POLITECNICO DI MILANO
Master of Science in Computer Science and Engineering
Department of Electronics, Information and Bioengineering

Imputation of biochemical activity associated with functional elements of the genome produced by epigenomic experiments

Laboratory of Data Science and Bioinformatics

Supervisor: Prof. Mark James Carman
Co-Supervisors: Prof. Marco Masseroli
Dr. Arif Canakoglu
Dr. Luca Nanni
Dr. Pietro Pinoli

Master Thesis of:
Francesco Guzzo, Matr 898035

Accademic Year 2018-2019

To my family...

Contents

List of Figures

List of Tables

Abstract

Epigenomics is the study of the modifications of the genome, i.e., the complete set of DNA which encodes all the information necessary for the functioning of an organism, that do not involve changes to the DNA sequence itself. Such modification includes chemical modification of the histones (the protein around which the DNA is packaged), DNA accessibility and interaction between proteins and DNA.
Nowadays, thanks to huge improvements made in genome sequencing techniques we have access to a vast amount of data that incorporates a variety of biochemical activities coming from the different genes. This information is critical in order to understand the different genetic variations and mutations that may play a role to identify particular diseases, their initial development and progression. Even though the sequencing is a very powerful technique, it has its downsides; indeed, performing experiments that measure the genomic features is expensive and technical challenges may prevent a comprehensive characterization of the genome in determinate settings.
To overcome this problem, a valuable method is the development of techniques capable of predicting the outcome of those experiments and imputing the biochemical activity in the form of signals which correspond to data elements representing the DNA behavior. By imputing signals it is possible to support existing experiments and predict the outcome of the ones not yet performed.
In this thesis we develop new algorithms able to impute signals with high precision and, taking as a baseline the previous works done in this research area, attempt to outperform them. The first part of the work, exploiting the data provided by the ENCODE Consortium, consists of data analysis, data pre-processing and the implementation of a Nearest Neighbour algorithm for signal imputation. The second part applies matrix factorization and tensor factorization techniques, in an attempt to improve performances and eventually extract potentially useful information from the trained models.

Sommario

L'epigenomica è la scienza che studia le modifiche del genoma, cioè il set completo di DNA che codifica l'informazione necessaria per il funzionamento di un organismo che non coinvolge cambiamenti della sua stessa sequenza. Questo tipo di cambiamenti include modifiche a livello chimico degli istoni (proteine intorno alle quali il DNA è avvolto), all'accessibilità al DNA e all'interazione tra proteine e DNA.
Al giorno d'oggi, grazie all'affinamento delle tecniche di sequenziamento genomico, abbiamo accesso ad un vasta quantità di dati che incorpora un grande varietà di attività biochimica proveniente dai geni. Questa informazione è fondamentale per comprendere a pieno le differenti mutazioni genetiche, le quali giocano un ruolo importante nell'identificazione di specifiche malattie, del loro esordio e sviluppo. Nonostante il sequenziamento sia una tecnica molto efficace, essa presenta degli svantaggi; infatti, eseguire esperimenti che misurano le *features* del genoma è estremamente dispendioso. Inoltre, numerose problematiche dal punto di vista tecnico potrebbero prevenire, sotto determinate condizioni, una caratterizzazione comprensiva del genoma.
Per superare questo ostacolo, una metodologia valida consiste nello sviluppo di tecniche in grado di predire il risultato di tali esperimenti ed imputare l'attività biochimica sotto forma di segnali, i quali costituiscono i dati che rappresentano l'attività del DNA. Infatti, attraverso i segnali imputati è possibile supportare i già esistenti esperimenti, predicendo l'esito di quelli non ancora effettuati.
All'interno di questa tesi, riportiamo lo sviluppo di nuovi algoritmi in grado di imputare segnali con un'alta precisione cercando di migliorare le *performance* ottenute da alcuni algoritmi implementati precedentemente in questo ambito.
La prima parte del lavoro, utilizzando i dati forniti dall'*ENCODE Consortium*, verte sull'analisi dei dati, il loro *pre-processing* e l'implementazione di un algoritmo per l'imputazione dei segnali basato sui *Nearest Neighbours*. La seconda parte, invece, approccia il problema attraverso tecniche di fattorizzazione per matrici e tensori, cercando di migliorare i risultati già ottenuti ed, eventualmente, estrarre dai modelli trainati informazioni potenzialmente utili.

Ringraziamenti

Ringrazio il mio relatore Mark James Carman e i miei corelatori Marco Masseroli, Arif Canakoglu, Luca Nanni e Pietro Pinoli per avermi seguito durante tutto il percorso della tesi.
Un ringraziamento speciale va alla mia famiglia che mi ha da sempre sostenuto e incoraggiato.
Un rigraziamento importante va a Sara che mi ha sopportato, reso felice in questi ulimi anni e aiutato in modo indispensabile durante la fine del mio percorso universitario.
Un particolare ringraziamento va a chi ha seguito con me e reso lieto questo percorso univeritario: il Consiglio dello Iench, Maffe, il Frys, Mazze, Ale, Alessia e Daniele.
Ringrazio anche Matteo, per l'importante supporto epigenetico, ed Enrico, Bruno, Andrea, Daniele, Attilio e tutti i miei amici senza i quali non sarei arrivato dove sono ora.

Chapter 1

Introduction

1.1 Research area

The term ”genome” refers to the complete set of DNA which encodes all the information necessary to build the entire organism, e.g., the human body. The units that compose the genome, which are therefore one of the principal elements of interest for the genomics, are the genes, traditionally seen as the regions of the DNA that carry the key information necessary to synthesize a specific protein or a set of proteins.
Genomics is the area of research that studies how the components of the humane genome work and interact between each other. Due to its critical importance, the studies related to this research area have evolved very rapidly across the last years, especially after the completion of the Humane Genome Project (HGP) in 2003, when scientists defined in detail the complete set of human genes with information about their structure, organization and function [1]. The amount of new information brought in by this revolutionary project and the possibility to access the blueprint of the structure and elements of the genome switched the attention of the studies toward the analysis of the specific roles of the different functional elements of the DNA, changing completely the meaning of the word ”gene”. Indeed, when speaking about the studies on genes (regions of the genome) it is no more correct to just refer to the analysis of regions required to encode proteins, in fact, the aim of the research has now as a target a broader variety of different functionalities related to the cellular life. More in depth, it is now important to be able to create a map of the different functional elements in the human genome, including: elements that regulate the activity of the genes and elements that act at the protein levels [2]. This type of research belongs to the field of epigenomics that studies the alterations of the genome that do not cause the modification of the original information embedded in the DNA, allowing to identify which are the regions that are ”actively working” and to concentrate in a targeted way the studies about the functionalities of those active genes.

Indeed, to map all the functional elements in the genome it is firstly important to identify them, then to focus on the role they play in the cellular life. According to this goal and studies, the National Human Genome Research Institute (NHGRI) founded the Encyclopedia of DNA elements (ENCODE) Consortium [2], which is an international collaboration, to investigate and identify, employing different epigenomic assays, functional elements in the genome. However, building a comprehensive map of the functional units and performing experiments that measure the genomic features is, in fact, expensive, time consuming and technical challenges may prevent the characterization of the genome in determinate settings (e.g., not all the cellular types or tissues are available to be harvested), so computer science methods integrate and support the research trying to solve those problems, giving valuable indications by predicting where potential areas of activity could occur across the genome. This thesis can then be located in a research field that benefits from a symbiotic relationship between epigenomics, which provides the raw data, and the computer science world, which processes and extracts from it all the relevant paths and information.

1.2 Goals and research applications

The raw data provided by the biological experiments, in order to be compatible with the computer science algorithms, is processed representing how likely there will be activity in a certain region of the genome for a certain cell or tissue, under a specific sequencing technique. The goal of the thesis is to implement an algorithm able to learn from the available data and predict with the best possible accuracy in which locations of the target genome there will be potentially active regions. Indeed, the imputation task of this work is carried on joining the *ENCODE Imputation Challenge*, with the aim of improving the performances of the previous state-of-the-art models. The main factor we will concentrate on is, in fact, to find a model that well fits the available data, introducing new ones or applying reasoned changes to the existing ones.
Other than performances, another possible objective is to enable for future works the possibility to extract meaningful information from the trained models in order to give a better characterization of the imputed biological activity.
Investigating the problem from a broader perspective, the ultimate long term goal is to take part in the research project that aims at completing the map of the functional elements of the genome.
From a biological point of view, the ability to understand the locus of the functional elements brings in important information that can be exploited to perform precise experiments and gain more knowledge about what all the genome functionalities are and how they affect the way cells

and the human body work. With the amount of data about human DNA, scientists have more powerful tools to investigate the role that multiple genetic factors, acting together and with the environment, play in complex diseases such as cancer. In fact, thanks to studies on the genome, medical researchers are able to develop more effective treatment, improved diagnostics, evidence-based approaches for demonstrating clinical efficacy, and better decision-making tools [1]. Ultimately, the research pace would be sped up by considerably if the algorithms developed prove to be robust and reliable, since manually producing experiments takes much longer than training a model and imputing target instances.

1.3 Brief description of the work

The first step of the thesis involved partecipating in the ENCODE Imputation challenge [2]. The challenge provided us the data and a period of time to work on an imputation algorithm and after that to submit our predictions. Along with the challenge data, we were provided with the results of the main baseline algorithm to compare with when trying to get good performance: namely Avocado [3], that exploits a multi-scale deep tensor factorization. Other than Avocado the other two baselines to look at when speaking about algorithm performance are ChromImpute [4] and PREDICTD [5]. The former trains an ensemble of regression trees. The latter exploits PARAFAC tensor decomposition in order to perform imputation. Each of the methods will be discussed more accurately in the following chapter.
The dataset is a tensor with three dimensions, the first one is composed by different types of cells and tissues, the second is composed by different assays that represent the techniques used to study the epigenomic state of the given sample and the third one represents the base pairs of the genome. Each value in the tensor reflects the outcome of the epigenetics technique related to the given cell assay and base pair triple (both structure, origin and meaning of the data will be analyzed more in depth in Chapters 2 and 3). During the period of the challenge, it was possible to just access train and validation data with the test set being released after the publication of the results.
To start to define the structure of the algorithm, the first step is to make the data feasible to be used in an efficient way: memory wise each one of the raw training samples is too heavy (around 11GB each) and, consequentially, analyzing them becomes very time consuming. To solve the problem a downsampling approach has been carried on. Moreover, in order to analyze the data, a subset of the genome has been sampled to represent in a compact way the meaningful information (this type of assumption also speeds up by lot the process and allows to get a more comprehensive view on the tensor). The aim of the data analysis is to understand both the

distribution of the values and the possible relations between different cells and assays. The latter have been explored further by computing the Pearson correlation between each sample, allowing to estimate the similarities between different cells and assays.
For what concerns the data normalization task, in order to smooth the values of the samples, different transformations have been applied. The data transformations and relative results will be discussed in Chapters 3, 6. After data preprocessing, the first model implemented is a Nearest Neighbour (kNN) regressor that exploits the above similarities across the assay and cell dimensions. The idea is to compute predictions employing the information carried by the same assay across different cells and same cell across different assays. Different merge techniques have been used in order to efficiently include into the final prediction information coming both from cells and assays. After performing imputations considering the whole genome length, we investigated methods for finding similarities at local resolutions, with the idea to try to predict independently consecutive sections of the sample, always exploiting the same kNN algorithm, aiming at including more information from the considered dimensions.
With another kNN approach we investigated whether or not rescaling the samples could have been a useful way to improve the performance. The idea behind the approach is to first try to predict the locus of the biochemical activity, then the scale of the values of the identified regions.
The last techniques implemented are three factorization models: an additive model, a matrix factorization model and a tensor factorization model. The objective is to catch, through different decompositions, key aspects of the dimensions that compose the dataset. The capability of successfully training the model gives the possibility to correctly characterize all the parameters and, consequentially, to enhance the quality of the predictions.

1.4 Outline of the thesis

The thesis is structured as follows:

- Chapter 2: Background analysis necessary to understand which are the biological fundamentals of the research area. Moreover, we present a more in depth explanation of what the baseline algorithms are and how they work, plus the presentations of the main methodologies used in this research and the different goals we wanted to achieve exploiting each one of them.

- Chapter 3: In this section we describe more precisely the Encode Imputation Challenge, including the main problems that the challenge brings, how to solve them and how the data is structured. Also, we address how the data analysis task has been performed, the information extracted from it and the data normalization problem.

- Chapter 4: In this chapter all the implemented techniques related to the kNN model will be formalized in detail

- Chapter 5: In this chapter additive, matrix factorization and tensor factorization models will be formalized in detail.

- Chapter 6: All the results achieved in and outside the challenge will be reported along with the modalities with which all the experiments have been carried out.

- Chapter 7: In this section will be included the conclusions of the thesis, explaining the relevancy of the work, plus we describe the possible future changes that can improve on what has already been done and that can contribute to push towards the ultimate goal of the epigenomic research area.

Chapter 2

Background

2.1 Biological background

The DNA is a molecule found inside a special area of the cell called the nucleus; it contains all hereditary instructions necessary to guarantee the correct functioning of all the activities inside the human body's cells [6]. The information inside the DNA is represented by a code composed of 4 different elements (chemical bases): adenine (A), cytosine (C), guanine (G), thymine (T). The order with which the bases are put in a the sequence of DNA determines the information available to construct and maintain the organism. The chemical bases pair up to form base pairs (A with T and C with G) which are the main component of what is called a nucleotide, the unit of the genome; multiple units are arranged into two long strands that form a double helix, the DNA (Figure 2.1).

The DNA molecule is packaged into thread-like structures called chromosomes (the chromosomes considered in this work are chr1 to chr22 and chrX). Each chromosome is made up of DNA wrapped many times around proteins called histones, forming nucleosomes that support its structure [8]. Genes correspond to regions within the chromosomes that can be considered the basic unit of heredity, [9] coding the key information necessary to guarantee the cells correct functionalities.
In order to transfer the information coded by the genes into practice, there are two main phases that need to be mentioned to correctly understand how the genome functional elements work, express themselves and how they can be characterized from an epigenomic point of view [10]:

- Transcription: is the first step of gene expression where the DNA double helix unwinds near the gene allowing the RNA to transcript the information coded inside.

- Translation: the information gathered from the DNA is decoded

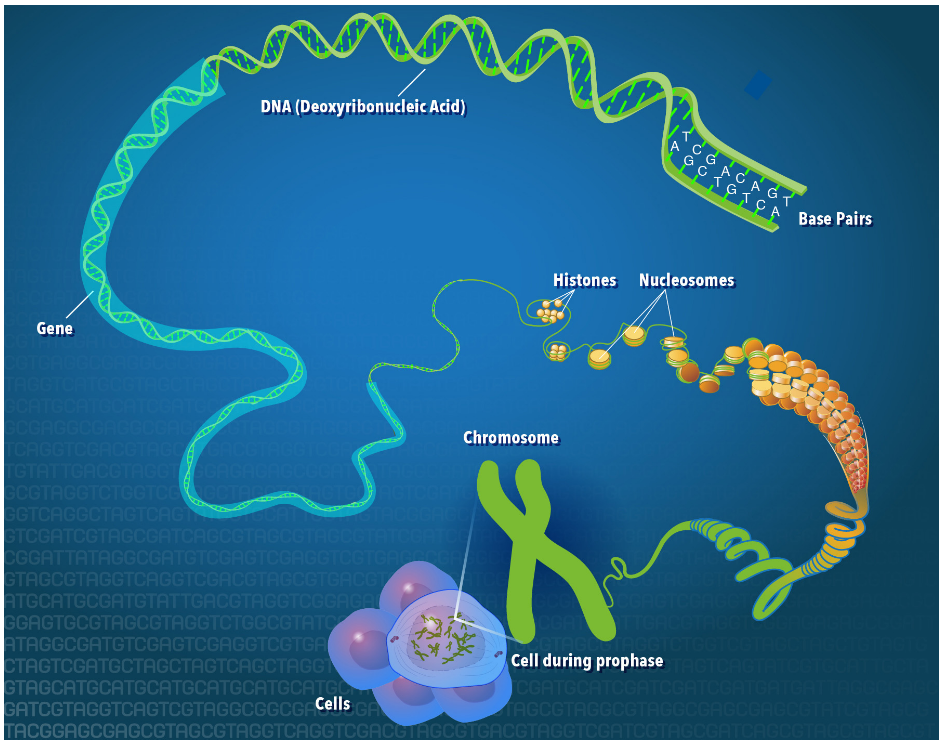


Figure 2.1: DNA double helix structure: from nucleotides to nucleosomes to chromosomes. Credit to [7]

in order to build proteins, which perform critical cellular functions, including the transcription itself and the regulation of the gene expression.

Having made clear the steps necessary to exploit the information inside the genes, it is now important to understand that the ones that keep instructions necessary to synthesize proteins correspond just to 1% of the genome; indeed, inside the 99% of the remaining regions, many sequences are functional and do not code for proteins but perform and code information critical for the cell; in particular, they regulate the gene's activity (e.g they can decide which gene must be active and which not across the chromosome) [11]. Major examples of non coding regions that perform regulatory activity are [12]:

- Promoters: Provide sites to bind the proteins apt to read information from the DNA. Typically, the promoter regions can be located ahead of the gene on the DNA strand.

- Enhancers: Provide binding sites for proteins that help activate the transcription of the information coded into the DNA. It is possible to locate the enhancers before or after the gene they are regulating, sometimes they can be also found far away.

This kind of variety of possible functionalities that can be discovered can play a key role in the process to understand in a comprehensive way how the genome works and it is, indeed, the main reason why this research area, and so the thesis, is pushing towards the complete mapping of the DNA functional elements.

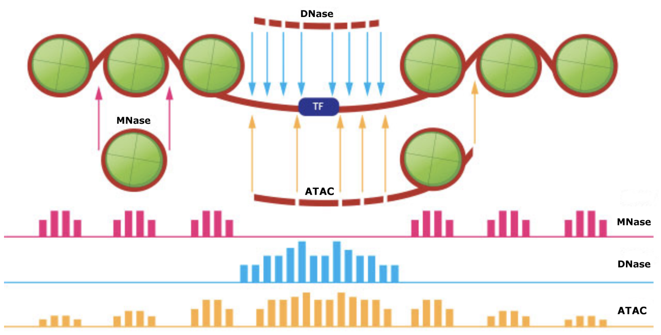
2.2 Epigenetics

The main way to study how these functionalities unfold across the genome is through epigenetics.
Indeed, epigenetics studies all the DNA modifications that do not change the DNA sequence but can affect gene activity [13]. There are, in fact, chemical compounds, entailing epigenetic changes, regulating both activity modifications for a given gene. These chemical compounds constitute the epigenome.
Epigenetics changes can help to understand if genes are active or not, therefore, can determine, for example, the production of certain proteins for a given cell, ensuring that only the necessary ones are produced. For instance, proteins that promote bone growth are not produced in brain cells [13]. Two common types of epigenetics modification, which are studied through out the assays composing the dateset exploited in this thesis, are:

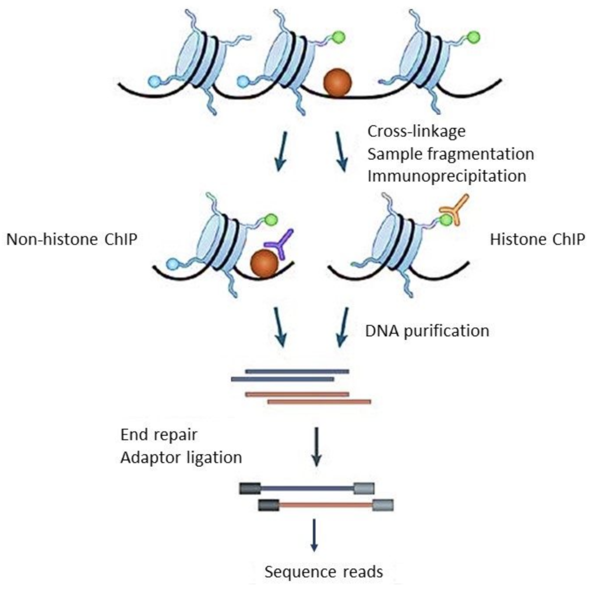
- DNA-Methylation: This process enables the turning off/silencing of a gene in order to inhibit the production of proteins. It unfolds when a methyl group is attached to the target segment of the DNA.

- DNA-Acetylation: This process increases the probability of nearby genes to be transcribed and works by adding an acetyl group to the target DNA segment.

The general technique that needs to be analyzed, in order to study the effects of those epigenetic marks and understand the activity of the functional units in the genome, is genome sequencing [14]. Sequencing simply means determining the exact order of the bases in a strand of DNA. The main idea behind the technique is to read mutiple segments of nucleotides in the DNA and assemble them through sequence overlapping in order to reconstruct a bigger section of DNA. The segment reconstruction can be interpreted as a picture of the current state of the genome. If this technique is paired with epigenetic modifications, the assay will allow to understand, under specific conditions, which are for example the regions where proteins interact with the DNA or where there is high chromatin accessibility. In particular, the sequencing techniques used to perform the assays [15] are the following (Figure 2.2):



(a) DNase-seq and ATAC-seq. Taken from [15]



(b) Chip-seq. Take from [16]

Figure 2.2: Sequencing techniques

- DNase-seq: Is used to understand which are the regions of the DNA that have chromatin accessibility, which are basically fragments of the genome that are accessible to regulatory proteins.

- ATAC-seq: Is used to investigate also the chromatin accessibility but with a different process. In this case, instead of the enzyme DNase-I, the sequencing is performed through an enzyme called Tn5 transposase. Both ATAC and DNase-seq result into a genome-wide track that represent the state of chromatine accessibility.

- Chip-seq: is used to obtain genome-wide profiles of specific histone modifications (the histone modification is defined as immunoprecipitation of the chromatine). Examples of histone modifications are methylation and acetylation and they basically allow to analyze fragment of DNA wrapped around the histone proteins.

2.3 Signals processing

Given a brief overview on the biological aspect of the thesis, is now important to understand the nature of the data on which we are working on and how the sequencing technique output is translated into a friendly format for a computer scientist [17]. Starting from the begin, all the sequences of DNA coming from the sequencing reads have the same length (e.g., 36 base pairs) and are, for the download and processing steps, encapsulated into a *tagAlign* format [15]. The first step is the subsampling, where all the reads, which are sequences of bases, are selected according to a certain depth, which in this case is 30 millions. The 30 millions reads are then aligned on the reference track of the genome GRch38, where, according to the position of the considered read, the segment is associated with the genome reference corresponding locus. The step is performed through the BWA software for alignment [18]. Subsequently, only the first position x on genome of each read is considered and extended in a window of 150 base pairs (x-75; x+75), where the number 150 corresponds to the average distance expressed in base pairs between two nucleosomes. The next step consists in defining a genome-wide pseudo background based on the number of counts available for each base pair, where the counts correspond to the number of instances showcased by the aligned reads on the genome (the reads can overlap, increasing the number of counts). Through a process called *Peak calling*, the significance of each number of counts for each base pair is defined with respect to the counts of the nearby base pairs with a Poisson distribution. The specific count, which is considered significant if a certain confidence level is met (the *p-value* needs to be less than a certain threshold), will express weather or not, in the corresponding region of the genome, there is evidence of a peak of biochemical activity. These last two steps are performed using the MACS2 software [19].
The values of the signals, for each base pair, are then obtained through the negative $\log_{10}$ of the Poisson p-value counts (negative $\log_{10}$ p-value

score provides a way to threshold the samples). The final tracks, which correspond to confidence scores, provide a measure of statistical significance of the observed biochemical activity in the form of genome-wide signals that correspond to each couple cell-assay of the dataset (Figure 2.3), where a dataset consists of signals generated from applying many assays to different cell-lines.

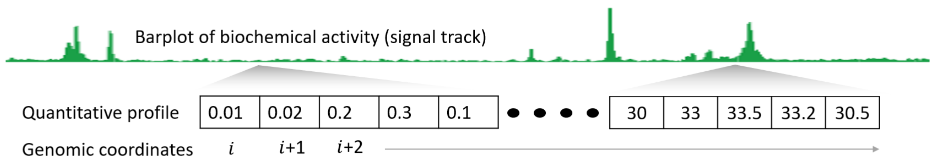


Figure 2.3: Example of signal track. Taken from [15]

2.4 Related works

There exist already a lot of studies about different techniques that, starting from the above generated signals, are feasible for the process of biochemical activity imputation. Three are the main ones that have been used as reference in this thesis: *ChromImpute* [4], *PREDICTD* [5], *Avocado* [3].

2.4.1 ChromImpute

The ChromImpute method consists in an ensemble regression-based approach to epigenomic imputation. The used dataset is composed by 127 different cells and tissues and 34 distinct assays. The largest part of the cells (111 samples) comes from the *Roadmap Epigenomics project*, 16 come from *ENCODE* Consortium. The 34 assays include 30 histone modification, DNA-methylation, DNase I hypersensitivity, RNA-seq and histone variant H2A.Z. The 34 genomic marks/assays are partitioned into 4 different tiers, each one of them includes a determined type of assay. The partition establishes which cell-assay signals can be used in order to predict a target track (e.g tier 1 assays are used just to impute tier 1 assays). Respecting the tier constraint, the algorithm main leverages on two different type of sample:

- Same sample different mark: considering the same cell line with different assays applied on.
- Different sample same mark information: considering the same experiment performed across different cells.

Each predictor is a tree that integrates two type of feature combined together: in order to predict a target assay j_t performed on a target cell i_t, one group of feature will come from same cell i_t related to different assay

tracks from j_t, the other one from same assay j_t but performed on different cells than i_t.
The tree, which integrates those features, is trained using each cell that has the target assay available. The average of all the predictors results will consists in the final prediction of the target signal Y_{ij}.

2.4.2 PREDICTD

The PREDICTD approach was applied on a dataset composed by 127 cells and 24 assays coming from the *Roadmap Epigenomics project.* The whole tracks at 25bp resoultion compose a 3D tensor with dimension $127 \times 24 \times BasePairs$. The method, which is based on the PARAFAC tensor decomposition, has two main group of parameters that need to be trained: the biases and the latent factors, respectively defined with three vectors and three matrices. Defined K, J, I respectively as the cardinality of cells, assay, base pairs and k, j, i as the specific cell, assay and base pair, the biases matrices c, a, g (cell, assay, genome) have dimensions $K \times 1$, $J \times 1$, $I \times 1$. Instead, the factor matrices C, A, G have dimensions $K \times L$, $J \times L$, $I \times L$ with L equal to the number of latent factors. The different latent factors are trained using stochastic gradient descent (SGD), by minimizing the squared error (SE) objective function:

$$argmin_{C,A,G,c,a,g} \sum_{j,k,i \in S^{train}} (D^{train}_{j,k,i} - [\sum_{l=1}^{L} C_{j,l} * A_{k,l} * G_{i,l} + c_j + a_k + g_i])^2 + \\ + \lambda_c \|C\|_2^2 + \lambda_a \|A\|_2^2 + \lambda_g \|G\|_2^2 \tag{2.1}$$

This objective function contains the Ridge regularization terms with one lambda for each group of parameters. The training of the model is partitioned into three steps where each one includes an hyper-parameter tuning phase. In the first training step the base pair latent factors are held frozen and just the cell and assay latent factors are updated; in this case just 0.01% of the 25 base pair resolution signal is considered. After the cell and assay latent factors have converged, the second step consists in training the base pairs latent factors. Lastly, all the parameters are carried through a comprehensive training phase. The training of the base pair latent factors is based on the adaptive moment estimation (ADAM) which is an extension to the stochastic gradient descent procedure to update model weights iteratively from the training data [20]. It adapts the parameter learning rates based on the average first moment and the average of the second moment of the gradients. None of the reported results for the model consider the whole signal at 25 base pair resolution but rather exploit a subsample of the genome called the ENCODE Pilot regions [21].

2.4.3 Avocado

The last considered algorithm, which is also the most recent, is Avocado. It is the algorithm used as a baseline in the ENCODE imputation challenge and, as claimed in the reference publication [3], outperforms the two previous approaches PREDICTD and ChromImpute. The dataset used to perform the imputation task is composed by 127 cells and 24 assays, part of a tensor of dimension $127 \times 24 \times GenomeLength$; the assays include 23 histone modifications and DNase sensitivity. The method consists in a multi-scale deep tensor factorization (Figure 2.4). The first step is the training of the latent factors which is, also in this case, performed trough the ADAM optimizer, partitioned in two different steps, and executed on just the ENCODE Pilot regions. First the cell and assay factors are trained, then are held frozen starting with the training of the base pair latent factors for each chromosome. Again the model is trained at last considering all the parameters.

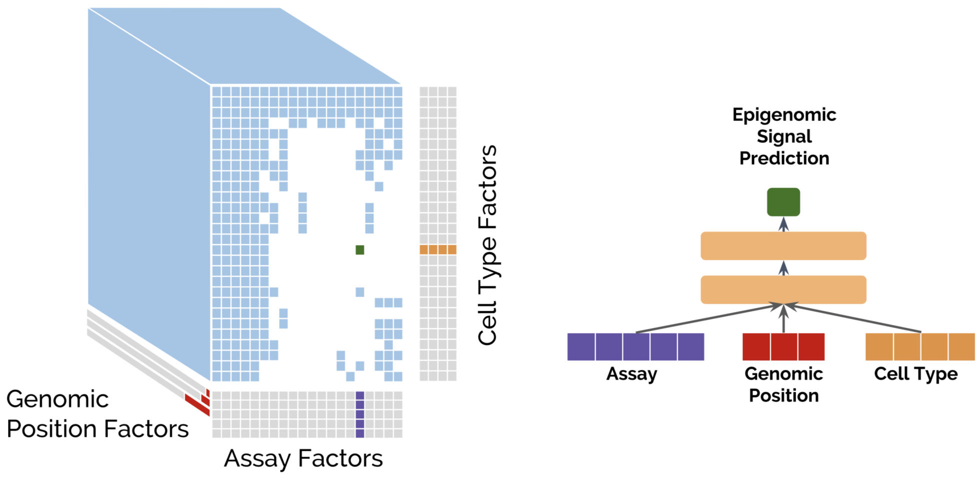


Figure 2.4: Multi scale deep tensor factorization. Credit to [3]

The two main differences between PREDICTD and Avocado are that, the former combines the latent factors related to cells, assays and base pairs in a straightforward way, linearly combining them with a generalized dot product, while the latter inserts a deep neural network (DNN) in place of the dot product for which the latent factors are concatenated in one input vector. Moreover, for what concerns the latent factors of the base pairs, instead of just using the 25 base pair (bp) resolution units of the genome, Avocado employs three different sets of resolution which are 25bp, 250bp and 5kbp. Finally, Avocado requires only $\sim$ 3.7 percent of the $\sim$ 92.2 billion parameters employed by PREDICTD's tensor factorization model, increasing the efficiency of the algorithm.

2.5 Metodologies

Here I will present from just a theoretical point of view the three main approaches that I have been used during the thesis to perform the imputation task.

2.5.1 Nearest neighbors regression

The k-nearest neighbors is an instance based approach, which constructs the predictions directly from the training instances themselves [22]. Indeed, there is no actual model to train in order to build the knowledge necessary for the predictions; the knowledge is constituted by the training set itself. This kind of models implement a *lazy learning* scheme which is an evaluation strategy that delays the evaluation of a new expression until its value is needed, avoiding repeated calculations.
Given an unknown test instance, a distance function is exploited in order to define which are the closest training samples to it; once the distances are defined and the closest samples are located, they are used to predict the class or the value of the test set sample. The two main aspects of the k-nearest neighbor approach are: finding the appropriate distance function for the dataset and the correct number k of neighbors necessary to get the best predictions. The most common distance measure used is the Euclidean distance but, in order to define how close two samples are one can use a similarity measure. In fact, similarity can be seen as an orthogonal yet related concept to distance (strong similarity means small distance and so the other way around). In this work, the similarity measure used, in order to compute how close two samples are, is the *Pearson correlation coefficient*:

$$\rho_{xy} = \frac{\sum_i^n (x_i - \overline{x}) * (y_i - \overline{y})}{\sqrt{\sum_i^n x_i^2} * \sqrt{\sum_i^n y_i^2}} \tag{2.2}$$

where n is equal to the number of samples, x_i is the i-th sample and $\overline{x}$ is equal to the average of the samples.
The number k of neighbors needs to be selected carefully, if k is too small, the prediction may be too sensitive to noisy points, if it is too big, the algorithm might end up including training samples too dissimilar between each other. The prediction of the algorithm can be related to both a classification or regression problem; the imputation process is carried on as a regression problem. The regression task involves, as the first step, the location of the set $\mathcal{K}$ of the k-nearest samples to the test instance; then, the prediction of the instance y can be performed as the average or the

weighted average of the nearest neighbors values:

$$y = \frac{\sum_{x \in \mathcal{K}} x}{|\mathcal{K}|} \tag{2.3}$$

$$y = \frac{\sum_{x \in \mathcal{K}} \rho_{xy} * x}{\sum_{(x,y):x \in \mathcal{K}} \rho_{xy}} \tag{2.4}$$

2.5.2 Matrix factorization

Given a matrix R $m \times n$ with rank $k \ll \min\{m, n\}$ is it always possible to express it as the product [23]:

$$R = UV^T \tag{2.5}$$

where the matrix U has dimension $m \times r$, the matrix V has dimension $n \times k$ and the rank of both row space and column space is k. Each column of the matrix U or V is referred as s *latent component* and each row of the matrix U or V is referred as a *latent factor*. Given $\overline{u_i}$ the i-th latent fator of matrix U and $\overline{v_j}$ the j-th latent factor of the matrix V, each element r_{ij} of the original matrix, as it follows from equation (2.5), can be expressed as:

$$r_{ij} = \overline{u_i} \cdot \overline{v_j} \tag{2.6}$$

Matrix factorization models are widely used in the recommender systems world [24] and it is, indeed, useful to contextualize the algorithm using a user rating matrix (URM), instead of a general R $m \times n$, in order to get a better intuition on what is the actual meaning of the decomposition and why it is valuable to factorize the input matrix.
So, an URM $m \times n$, with each row i corresponding to a general user, each column j corresponding to a general item and with each element (i, j) corresponding to the rating an user i has given to an item j, can be factorized as the product of a $m \times r$ matrix U and a $n \times r$ matrix V with r being the number of latent factors. The values in the matrix U represents the value of affinity that a certain user i has towards certain characteristics of an item j and the matrix V represents the membership value of a certain item j for certain characteristics. It is then more likely that, if a user affinity matches the item membership, the corresponding rating value (i, j) will be higher. Exploiting this kind of model one can express the predicted rating r_{ij} as:

$$r_{ij} = \sum_{l=1}^{k} u_{il} * v_{jl} \tag{2.7}$$

In order to define which are the parameters necessary to define the matrices U and V, it is possible to formulate an optimization problem as:

$$\min J = \frac{1}{2} ||R - UV^T||^2 \tag{2.8}$$

The problem can be solved by training the parameters with a stochastic gradient descent algorithm (SGD [25]) and once the objective function is minimized, it will be possible to impute all the missing values of the original matrix R (if it is a sparse matrix).

2.5.3 Tensor factorization

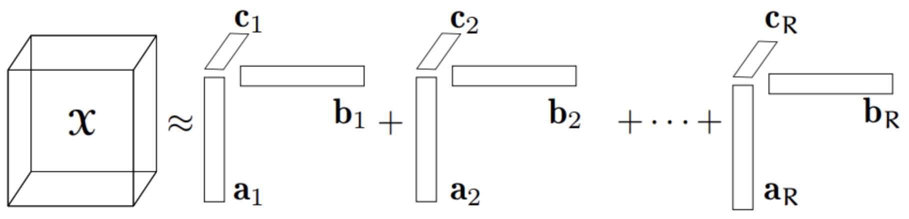


Figure 2.5: Tensor factorization

A tensor is a multidimensional array. More formally, an Nth-order tensor is an element of the tensor product of N vector spaces, each one with its own coordinate system [26]. The most general form of tensor decomposition is the canonical polyadic decomposition (CPD) or PARAFAC decomposition (Figure 2.5), which is also the one we employed in this work. The key concept of this rank decomposition is to express the tensor as the sum of a finite number of rank-one tensors [27].
Given a third order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ defined L as the number of latent factors used for the decomposition, it can written as:

$$\mathcal{X} = \sum_{l=1}^{L} a_l \cdot b_l \cdot c_l \tag{2.9}$$

where $a_l \in \mathbb{R}^I$, $b_l \in \mathbb{R}^J$ and $c_l \in \mathbb{R}^K$ are the rank one components of the decomposed tensor.
The number of rank one components corresponds to the cardinality of the tensor and will be composed by a number of elements equal to the number of latent factors. In order to define a single element of the tensor x_{ijk} the element-wise corresponding formulation is:

$$x_{ijk} = \sum_{l=1}^{L} a_{il} * b_{jl} * c_{kl} \tag{2.10}$$

Also in this case, the tensor decomposition, as the matrix factorization algorithm, results very valuable in order to express the different elements across the three dimensions, expressing them through characteristics integrated inside the trained latent factors parameters. The training of the basic PARAFAC decomposition model can be performed with the minimization of an objective function. An example is the one previously formalized for the state-of-the-art model PREDICTD (2.1).

2.6 Research questions

Given that the ultimate goal of the thesis is to perform the imputation task with the best possible results, trying to outperform the above baseline predictors (in particular the Avocado one), in this last section of the background I will report which are the the main research questions and expected goals that lead us to the selection of the aforementioned methodologies. The main and more general question concerns the way in which we could exploit those methods in order to improve over the selected baselines. Taking as an example the KNN regressor, from both the theoretical and implementation point of view, the method is simpler than a deep tensor factorization and we decided to commit to it for two main reasons: (i) we wanted to investigate something relatively novel in the research area (ii) we wanted to understand if in harsh conditions like high sparsity, low number of training data and challenging distribution of the test set (the dataset will be carefully described in the following chapter), where training a model in order to achieve good performances can result at times very difficult, a basic instance based method, which naturally extracts the knowledge directly from the data, could be able to achieve similar or even greater performances. Indeed, all previous state-of-the-art methods worked training from datasets with lower sparsity, ensuring that the distribution of the test data with respect to the the training data was such to allow a more efficient training of the model. Thus, the task of comparing them with the kNN approach could turn out to be interesting and productive from a research point of view, providing a new possibility when there is the necessity to impute for dataset under the above conditions.
The second adopted approach is the Matrix factorization (which can be considered a 'superset' of the additive model that will be also presented in Chapter 5). Also in this case, the idea is to introduce an interesting term of comparison for the tensor factorization models. As will be further explained in the related Chapter 5, the idea behind the implemented algorithm is to perform factorization of the cell and assay dimension multiple times across the genome. Doing so, we wanted to better comprehend if it was actually necessary to adopt the tensor decomposition model, factorizing also across the genome dimension, or it is actually possible to impute independently the value at each base pair from just training the characteristics/latent features of the corresponding cell and assay dimension, achieving similar or even better performance.
Finally, we decided to employ the tensor factorization method based on what both the Avocado and PREDICTD method introduced in their ap-

proach. We tried to showcase if the way in which their training task has been carried out can actually lead to the best possible predictions. The main question, specially for what concerns the PREDICTD algorithm, is the choice to implement as biases three matrices c, a, g with dimensions $K \times 1$, $J \times 1$, $I \times 1$. Indeed, considering the fact that they trained their model with over 1000 different samples, the idea of summarizing the basic characteristics of the genome into one single vector g, differentiating the predictions by just summing the same constant values relative to the current couple cell-assay across all the positions, can actually bring to the loss of potentially useful information that can lead to achieve even better performance. We did, in fact, try to understand if it is possible to summarize more efficiently and with more accuracy the information of the dataset (always with the PARAFAC decomposition) by structuring differently the biases, considering one genome-wide vector for each cell and each assay. Also, given the high number of latent factors in the PREDICTED model, we wanted to understand weather or not adding to Ridge the Lasso regularization (Elastic net regularization [28]) term could have brought to a natural selection of the most important parameters in the model, improving performances but also allowing an easier interpretation of the results. Indeed, a more interpretable trained model could potentially lead to a better understanding of important related biological features.
Instead, for what concerns the training phase of Avocado and PREDICTD, the general idea of training separately the cell and assay from the base pairs latent factors and then train them together can eventually lead the model to learn for some parameters sub-optimal information. The algorithm, in the first two steps, is, in fact, trying to optimize the objective function updating the assay and cell factor considering the base pairs latent factors with their original initialization (which does not represent any kind of information of the data). So, we questioned weather or not could have been better to train all the parameters concurrently, stopping the model from learning eventual 'false positive' information prior to their final comprehensive training.
Here is an outline to better summarize the research questions that brought us to employ the different methods with the relative adjustments: enumitem

(RQ_1) Given the structure of the available data, is it possible to outperform factorization models with a kNN approach?

(RQ_2) Can factorizing each element of the genome based on just cell and assay with the matrix factorization model achieve good performances and, consequentially, introduce an interesting term of comparison for the previous baselines?

(RQ_3) Is it possible to achieve more optimal solutions with the tensor factorization model by adopting a new structure to summarize information into biases?

(RQ_4) Will training concurrently all the parameters and adopting the elastic net regularization term for the base pair factors lead to better performance and interpretability of the model?

The description of the implemented models, their comparison and the answers to the above research questions will be provided in the following chapters.

Chapter 3

Investigating the data

3.1 ENCODE imputation challenge

The Encyclopedia of DNA Elements (ENCODE) Consortium is an ongoing collaboration of research groups founded by the National Human Genome Research Institute (NHGRI) [29]. The goal of ENCODE is to build a comprehensive parts list of putative functional elements in the human genome in order to lead towards a better understanding of: (i) basic genome biology and (ii) the molecular and genetic basis of different diseases.
Since, as already stated in the introduction (Chapter 1), performing assays on different cells or tissues is expensive the imputation of the biochemical activity of the different parts of the genome in the form of signal vectors is very valuable for biologists.
In order to push for further improvements of the quality of the predictions generated with computer science imputation methods, the ENCODE Consortium announced a challenge, *The Encode Imputation Challenge*, recruiting researchers to develop a new algorithm able to impute the proposed test set tracks, outperforming the current state of the art *Avocado* (Section 2.4).
The research problem of the thesis starts from joining the challenge and, exploiting the provided data and baselines performances (Average predictor and *Avocado*), evolves trying to improve the developed algorithms performances, even after the deadline imposed by the competition (15th of August 2019).

3.1.1 Data structure

The data provided with the challenge comes form the ENCODE Consortium and can be summarized with a three dimensional tensor with two short axes and one long axis (Figure 5.3). The short axes represent 51 different cells or tissues on which the experiments have been perfomed and 35 different genomic assays including DNA-seq, ATAC-seq and Chip-seq (Histone modification profiling experiments).

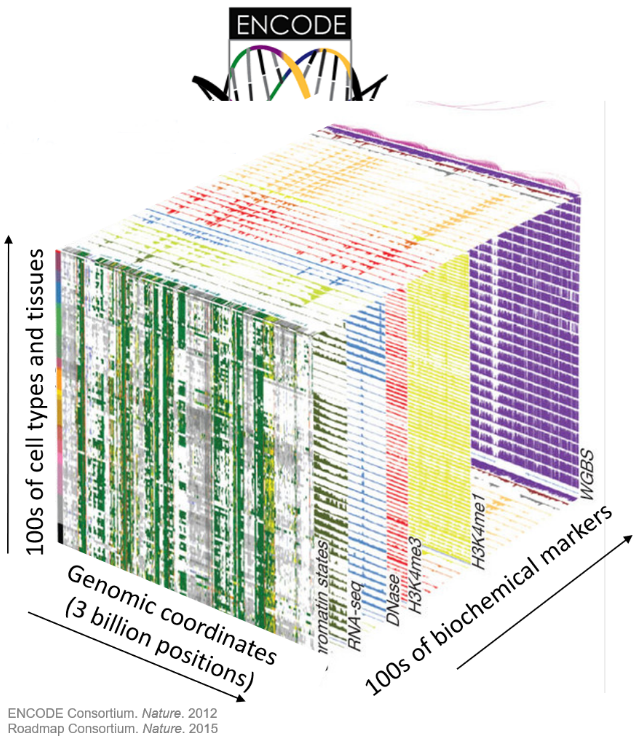
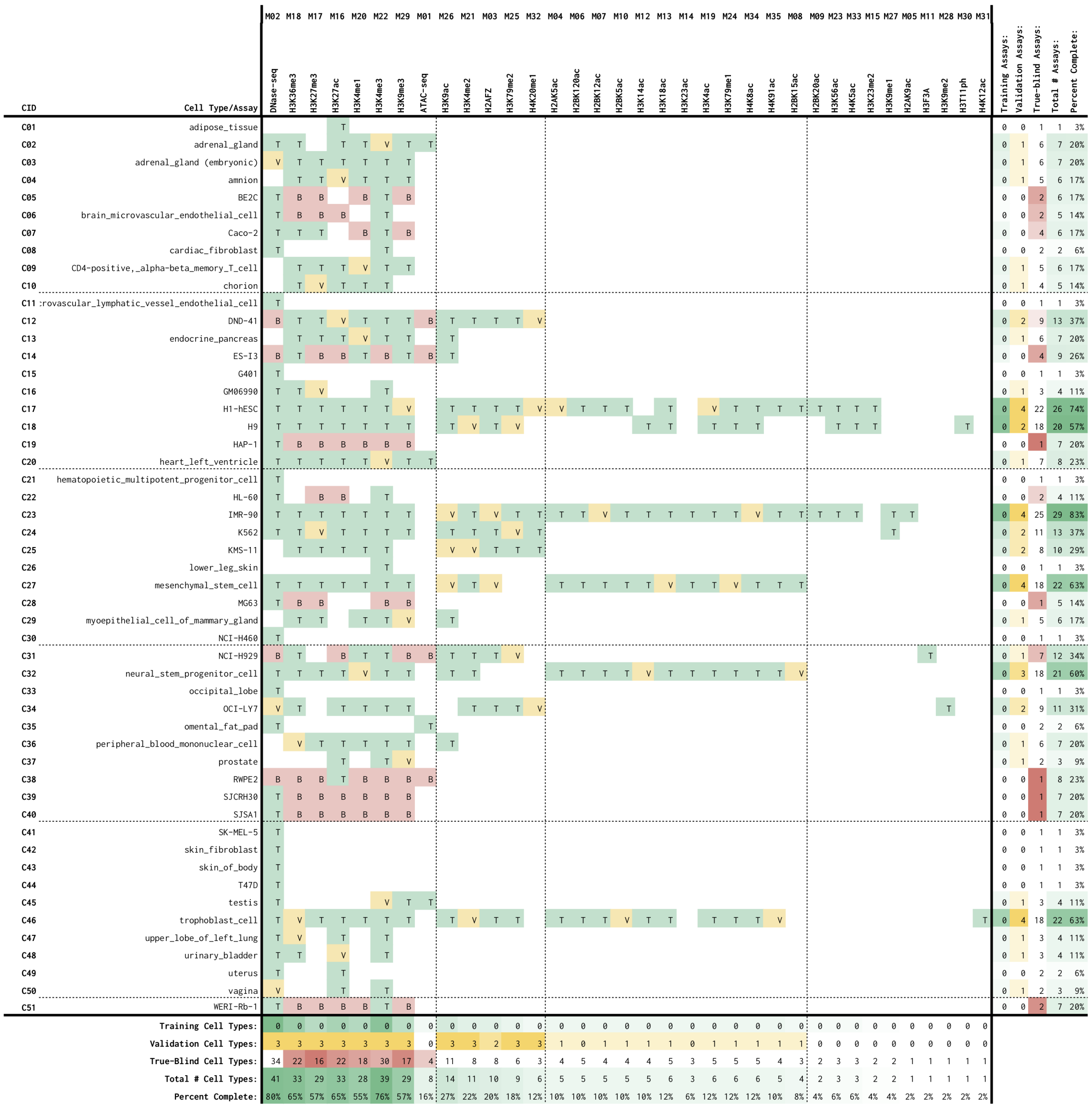

ENCODE
100s of cell types and tissues
Genomic coordinates (3 billion positions)
100s of biochemical markers
Chromatin states
RNA-seq
DNase
H3K4me3
H3K4me1
WGBS
ENCODE Consortium. Nature. 2012
Roadmap Consortium. Nature. 2015

Figure 3.1: Tensor data. Taken from [15]

The long axis has a total of 3.031.042.417 elements/base pairs that correspond to almost the entire human genome; the chromosomes involved are chr1 to chr22 and chrX, excluding chrY and chrM.
Each element of the tensor identifies the evidence of biochemical activity on a base pair for a given couple of cell and assay. The data cube can be visualized in a more intuitive way through the metadata table (Figure 3.2) that gives a better understanding on how the problem should be interpreted and approached.
Indeed, each element in the metadata table identifies a genome-wide signal, which is the outcome of its corresponding assay performed on its corresponding cell (Section 2.3). The data is composed by a total of 368 signals that, considering the entirety of the dimensions fill the 20% of the table.
The data is provided already split into training T and test B set with respectively 303 and 56 tracks. The former, also through hold out technique, is split again into 267 training T and 45 validation tracks V. Each signal in the data cube is provided as *bigWig* file, an indexed binary format designed to very efficiently store epigenomic tracks, which main advantage is allowing the representations of dense, continuous data in the genome browser as a graph [30]. Indeed, this format allows to access just the meaningful regions of the tracks and since the largest part of the signal is



CID	Cell Type/Assay	M02 DNase-seq	M18 H3K36me3	M17 H3K27me3	M16 H3K27ac	M20 H3K4me1	M22 H3K4me3	M29 H3K9me3	M01 ATAC-seq	M26 H3K9ac	M21 H3K4me2	M03 H2AFZ	M25 H3K79me2	M32 H4K20me1	M04 H2AK5ac	M06 H2BK120ac	M07 H2BK12ac	M10 H2BK5ac	M12 H3K14ac	M13 H3K18ac	M14 H3K23ac	M19 H3K4ac	M24 H3K79me1	M34 H4K8ac	M35 H4K91ac	M08 H2BK15ac	M09 H2BK20ac	M23 H3K56ac	M33 H4K5ac	M15 H3K23me2	M27 H3K9me1	M05 H2AK9ac	M11 H3F3A	M28 H3K9me2	M30 H3T11ph	M31 H4K12ac	Training Assays:	Validation Assays:	True-blind Assays:	Total # Assays:	Percent Complete:
C01	adipose_tissue				T																																0	0	1	1	3%
C02	adrenal_gland	T	T		T	T	V	T	T																												0	1	6	7	20%
C03	adrenal_gland (embryonic)	V	T	T	T	T	T	T																													0	1	6	7	20%
C04	amnion		T	T	V	T	T	T																													0	1	5	6	17%
C05	BE2C	T	B	B		B	T	B																													0	0	2	6	17%
C06	brain_microvascular_endothelial_cell	T	B	B	B		T																														0	0	2	5	14%
C07	Caco-2	T	T	T		B	T	B																													0	0	4	6	17%
C08	cardiac_fibroblast	T					T																														0	0	2	2	6%
C09	CD4-positive,_alpha-beta_memory_T_cell		T	T	T	V	T	T																													0	1	5	6	17%
C10	chorion		T	V	T	T	T																														0	1	4	5	14%
C11	:rovascular_lymphatic_vessel_endothelial_cell	T																																			0	0	1	1	3%
C12	DND-41	B	T	T	V	T	T	T	B	T	T	T	T	V																							0	2	9	13	37%
C13	endocrine_pancreas		T	T	T	V	T	T		T																											0	1	6	7	20%
C14	ES-I3	B	T	B	B	T	B	T	B	T																											0	0	4	9	26%
C15	G401	T																																			0	0	1	1	3%
C16	GM06990	T	T	V			T																														0	1	3	4	11%
C17	H1-hESC	T	T	T	T	T	T	V		T	T	T	T	V	V	T	T	T		T		V	T	T	T	T	T	T	T	T							0	4	22	26	74%
C18	H9	T	T	T	T	T	T	T		T	V	T	V						T	T		T	T	T				T	T	T					T		0	2	18	20	57%
C19	HAP-1	T	B	B	B	B	B	B																													0	0	1	7	20%
C20	heart_left_ventricle	T	T	T	T	T	V	T	T																												0	1	7	8	23%
C21	hematopoietic_multipotent_progenitor_cell	T																																			0	0	1	1	3%
C22	HL-60	T		B	B		T																														0	0	2	4	11%
C23	IMR-90	T	T	T	T	T	T	T		V	T	V	T	T	T	T	V	T	T	T	T	T	T	V	T	T	T	T	T		T	T					0	4	25	29	83%
C24	K562	T	T	V	T	T	T	T		T	T	T	V	T																	T						0	2	11	13	37%
C25	KMS-11		T	T	T	T	T			V	V	T	T	T																							0	2	8	10	29%
C26	lower_leg_skin						T																														0	0	1	1	3%
C27	mesenchymal_stem_cell	T	T	T	T	T	T	T		V	T	V			T	T	T	T	T	V	T	T	V	T	T	T											0	4	18	22	63%
C28	MG63	T	B	B			B	B																													0	0	1	5	14%
C29	myoepithelial_cell_of_mammary_gland		T	T		T	T	V		T																											0	1	5	6	17%
C30	NCI-H460	T																																			0	0	1	1	3%
C31	NCI-H929	B	T		B	T	T	B	B	T	T	T	V																				T				0	1	7	12	34%
C32	neural_stem_progenitor_cell	T	T	T	T	V	T	T		T	T				T	T	T	T	V	T	T	T	T	T	T	V											0	3	18	21	60%
C33	occipital_lobe	T																																			0	0	1	1	3%
C34	OCI-LY7	V	T		T	T	T	T			T	T	T	V																				T			0	2	9	11	31%
C35	omental_fat_pad	T							T																												0	0	2	2	6%
C36	peripheral_blood_mononuclear_cell		V	T	T	T	T	T		T																											0	1	6	7	20%
C37	prostate				T		T	V																													0	1	2	3	9%
C38	RWPE2	B	B	B	T	B	B	B	B																												0	0	1	8	23%
C39	SJCRH30	T	B	B	B	B	B	B																													0	0	1	7	20%
C40	SJSA1	T	B	B	B	B	B	B																													0	0	1	7	20%
C41	SK-MEL-5	T																																			0	0	1	1	3%
C42	skin_fibroblast	T																																			0	0	1	1	3%
C43	skin_of_body	T																																			0	0	1	1	3%
C44	T47D	T																																			0	0	1	1	3%
C45	testis	T					V	T	T																												0	1	3	4	11%
C46	trophoblast_cell	T	V	T	T	T	T	T		T	V	T	T		T	T	T	V	T	T		T	T	T	V											T	0	4	18	22	63%
C47	upper_lobe_of_left_lung	T	V		T		T																														0	1	3	4	11%
C48	urinary_bladder	T	T		V		T																														0	1	3	4	11%
C49	uterus	T			T																																0	0	2	2	6%
C50	vagina	V			T		T																														0	1	2	3	9%
C51	WERI-Rb-1	T	B	B	B	B	T	B																													0	0	2	7	20%
	Training Cell Types:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
	Validation Cell Types:	3	3	3	3	3	3	3	0	3	3	2	3	3	1	0	1	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0					
	True-Blind Cell Types:	34	22	16	22	18	30	17	4	11	8	8	6	3	4	5	4	4	4	5	3	5	5	5	4	3	2	3	3	2	2	1	1	1	1	1					
	Total # Cell Types:	41	33	29	33	28	39	29	8	14	11	10	9	6	5	5	5	5	5	6	3	6	6	6	5	4	2	3	3	2	2	1	1	1	1	1					
	Percent Complete:	80%	65%	57%	65%	55%	76%	57%	16%	27%	22%	20%	18%	12%	10%	10%	10%	10%	10%	12%	6%	12%	12%	12%	10%	8%	4%	6%	6%	4%	4%	2%	2%	2%	2%	2%					

Figure 3.2: *Flat data cube. Taken from [15]*

mainly composed by zeros, allows to focus the representation only where the actual biochemical activity is present along the genome.

3.2 Explorative data analysis

The first step, necessary to get a good grasp of the problem, is to explore the data trying to extract as much knowledge as possible, in order to have a better understanding of how the signal imputation should be structured and performed. Before getting into a more specific investigation of the the signals, it is beneficial to get a general overview on what are the main difficulties that can come along with the data provided by the *Imputation Challenge*. In this regard, three main topics are worthy to be presented: *Data format*, *Data size* and *Sparsity of the data*.

Data format

As already mentioned above, the *bigWig* files are very efficient from a storage point of view but they are slow for operations that require rapid repeated access to small chunks of the signal. The first step is then to unpack them into a more friendly format as provided by *numpy*. The process of unpacking consists of different steps. In the first step the *bigWig* files are transformed into a *BedGraph* file through the tool *bigWigToBedGraph* available in the USCS Genome Browser [30]. *BedGraph* files are line-oriented formats (Figure 3.3) that represent for each chromosome the value associated to the specified range of base pairs, *chromStartA* to *chromEndB*. The signals in the *BedGraph* format are then transformed into numpy arrays.

```
chromA  chromStartA  chromEndA  dataValueA
chromB  chromStartB  chromEndB  dataValueB
```

Figure 3.3: BedGraph format

Data size

Each signal, after unpacking, is represented as a numpy vector with dimension $1 \times$ *Number of basepairs* (~ 3 *billion*) which corresponds to an huge number of values. Indeed, a single signal is on average $10GB$, making it difficult to be easily managed. Computationally, standard tasks require too much time in order to make the imputation problem feasible. Moreover, the whole tensor weights $303 \times$ *10GB* and, in order to fit in memory, would require to be partitioned in smaller chunks that have to be loaded and stored many times to access the whole dataset, creating a bottleneck for every operation and algorithm we try to implement.
Thus, in order to make the data more manageable the signals are downsampled to a 25 base pair resolution. Each bin of the signals corresponds to the average value over 25 consecutive base pairs in the original track (from now on, every reference to the genome length will consider the 25 base pair resolution). This process drastically reduces computational time and allows to load without any problem more than one signal into memory. In order to work in an even more efficient way, more types of sampling will be presented in the thesis, each one suitable to carry out a specific task.

Moreover, the 25 base pair resolution signal is also the required format for the *Imputation Challenge* submissions.

Sparsity of the data

The high percentage of missing values (80%) frustrates the implementation of heavy training based techniques. This is why it is necessary to carefully chose the number of parameters of our models, especially for what concerns the number or latent factors for matrix and tensor factorization.
Moreover, the sparsity comes with the challenging distribution provided by the ENCODE team of training, validation and test set tracks. The test tracks (Figure 3.2) are grouped along specific cell-lines for which we have basically no information. Indeed, there is often just one training signal track, beside the test tracks, giving information about a particular cell-line. The choice of this type of distribution adds further difficulty and pushes for the development of an algorithm robust enough to work under these conditions. This is in fact one of the novel factors introduced by the challenge since all the previous algorithms were instead making sure to train the model with a much larger minimum number of sample for each cell/assay-line dimension.

3.2.1 Data visualization

After understanding which are the most challenging aspects related to the data, the next step is to perform a more in depth investigation of the training data itself, trying to get information useful to understand which algorithm to implement and how to structure it properly. In order to visualize how the different signals in the tensor are structured we used *violin plots*. These plots show and allow to compare the kernel density estimation of the underlying distribution of different variables. In this case, they allow us to compare the distribution of values for a group of designed signals.
The first step consists of showing the values that compose the different signals, trying to put a focus on the ones meaningful for the imputation task. Taking as example the 6 train signals from the cell line *C03* of the dataset, it is clear from the distributions in the plot (Figure 3.4) that the majority of the values are zeros. Since what we are trying to impute are the values that showcase biochemical activity in specific regions across the genome, in order to get meaningful visualizations, it is better just to focus on the top1% values of each signal. This should allow us to catch the relevant differences between tracks considering different cell assay couples (Figure 3.5).

From now on, for the following visualization I will take into consideration just the top1% of the signals values. The main thing to be investigated is weather or not the differences between the signals are strongly correlated

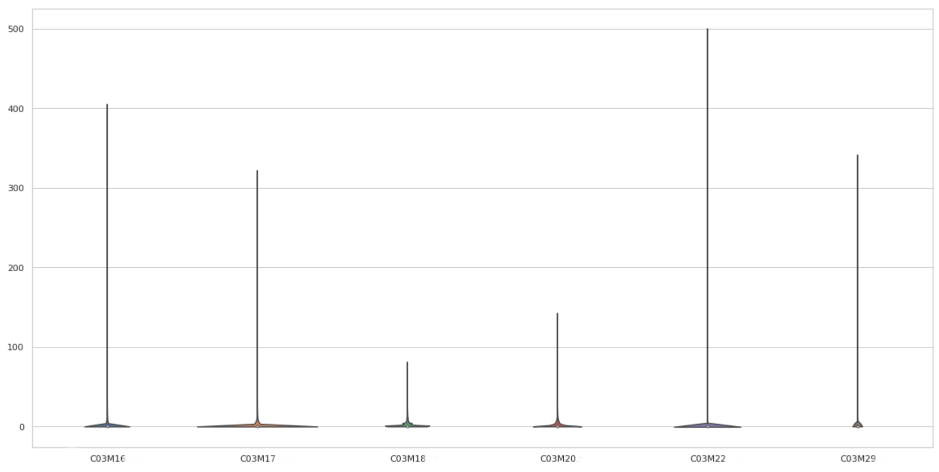


Figure 3.4: C03 signals distribution

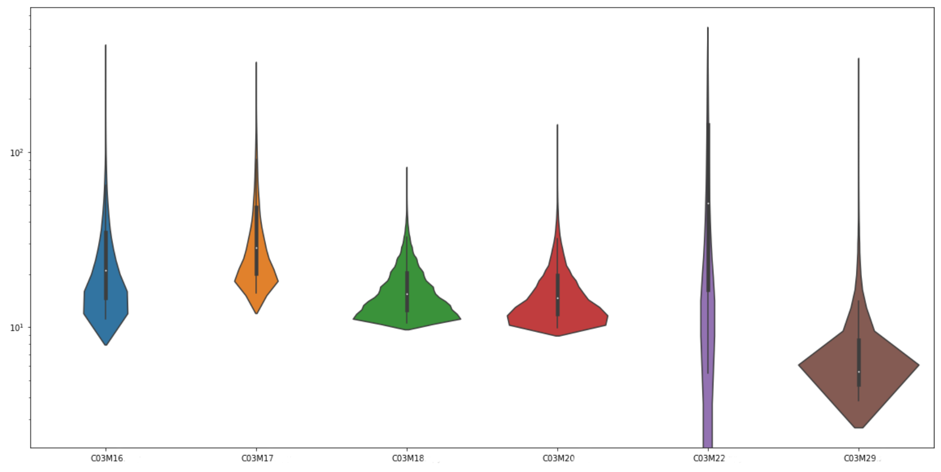


Figure 3.5: top1% C03 signals distribution with logarithmic scale

to the intrinsic features of their biological components, cells and assays. Indeed, a feasible way to perform this kind of analysis is to understand how the average values vary across all the different cells and all the different assays. In order to do this, considering the top1% signals, two measures have been exploited to characterize the different signals, the *median* (because it is much less biased towards outliers than the *average*) and *standard deviation*. So, after summarizing each top1% signal with the median value, we characterize each cell-line through a tuple *(mean, std)* where: (i) the *mean* corresponds to the average of all the median values of the signals belonging to that specific cell-line (ii) the *std* corresponds to the standard deviation of all the median values of the signals belonging to that specific cell-line. Once obtained, the tuples for each cell are represented through out the barplot in Figure 3.6.

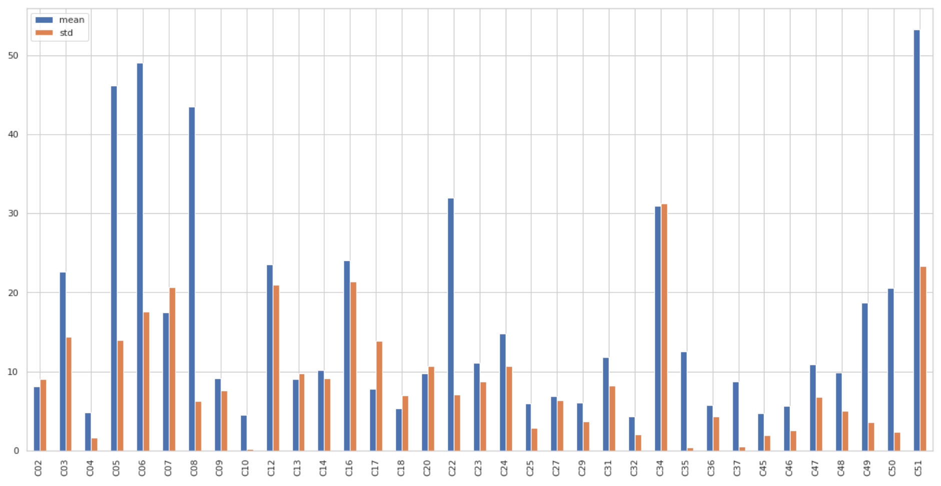


Figure 3.6: Mean and std of the median values for each cell

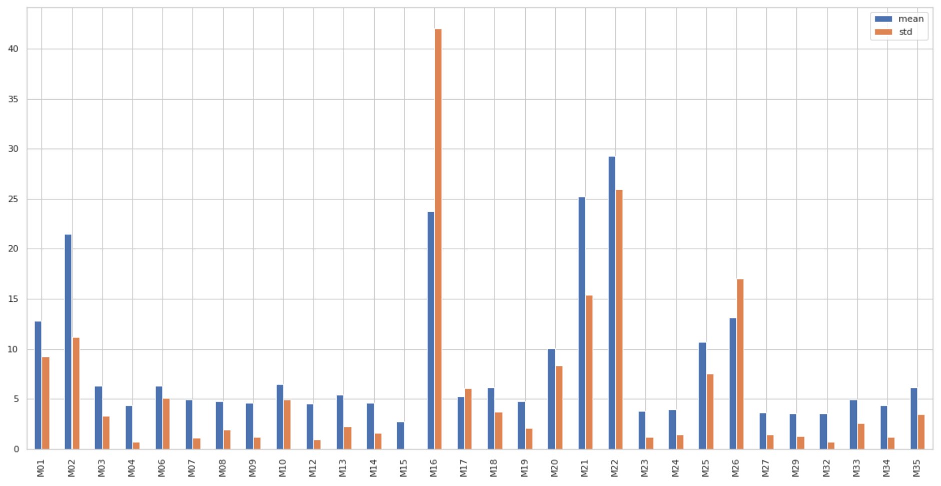
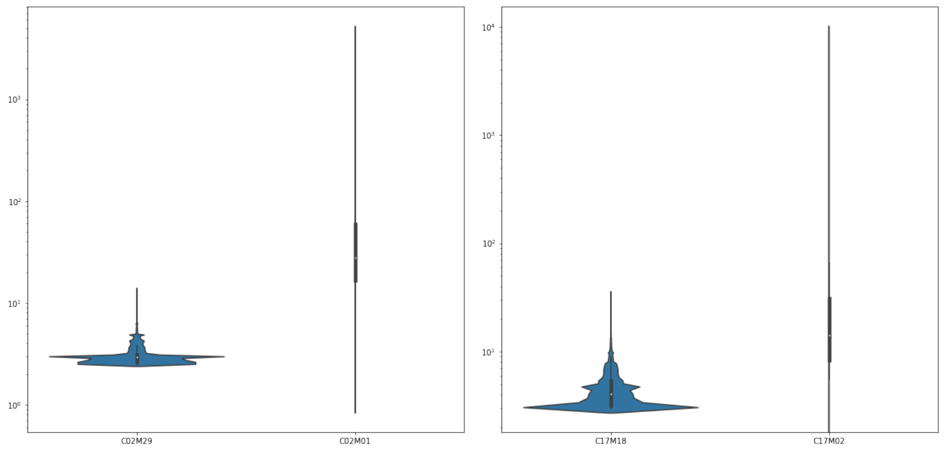


Figure 3.7: Mean and std of the median values for each assay

The same barplot has been extracted from the different assay lines (Figure 3.7) in order to confront the two different dimensions of the tensor.

The two visualizations show that the mean of standard deviation of the median value varies more across the different cell-lines than across the different assays. This suggests that the regions of activity and their corresponding values should be more similar for signals related to the same assay performed on different cells than for different assays applied on the same cell-line. Thus, it is possible to assume that, the particular technique used to generate the signal has a more important role when trying to discriminate the different tracks. Going more in depth, this kind of assumption can get further support from the comparisons of the distributions of the signals related to the different assay groups *(DNase-seq M02, ATAC-seq M01)* and histone modifications. Indeed, given two different cell-line *C02, C17* and, for each one, two signals related respectively to an histone modification and *ATAC-seq* or *DNase-seq*, it is clear how the difference in scale between the top1% values of the related tracks can be massive (Figures 3.8).



(a) C02 Chip-seq and ATAC-seq (M01) (b) C17 Chip-seq and DNase-seq (M02)

Figure 3.8: (DNase-seq, ATAC-seq) against Chip-seq top1% scales with logarithmic scale

3.2.2 Data outliers

It might be useful in the data investigation process to analyze the outliers of the the different signals.
We have seen that there are meaningful differences in scale between the *DNase, ATAC-seq* and other assays but the majority of the experiments in the dataset are *Chip-seqs* so it might be useful to analyze the outliers in the data from a more general perspective, without just focusing on two of the 35 assays. Indeed, in case the discrepancy in scale is too big between the different signals, it could be possible to rescale/normalize the dataset and avoid values that could represent a problem for the imputation problems.
Big outliers may not have an important meaning from a biological point of view, so predicting them wont be useful. From just a performance point of view, those values could end up biasing the predictions on the validation and test sets. It is, indeed, valuable to investigate those values and a possible way to do it is, always focusing on the top1%, to select, for each

signal, the average of all the values greater than $Q_3 + 1.5 * IQR$, where Q_3 is the third quartile and IQR is the difference between the third quartile and the first quartile (this should provide us information about the average outliers values in the dataset).

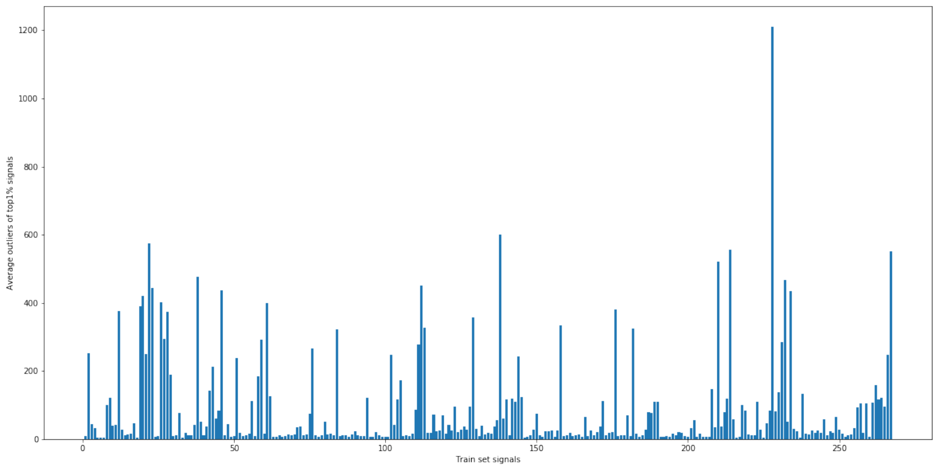


Figure 3.9: Average outliers value of top1% signals

Indeed, from Figure 3.9 it is clear how across the 267 signals of the training set there are remarkable differences in the highest values of the signals. So, in order to solve this problem and improve the quality of the predictions, the dataset has been transformed as will be explained in the following section.

3.3 Signal pre-processing

Other than the big differences due to the different nature of the experiments we have seen how, also considering all the signals of the training set, there are meaningful discrepancies in scale. Those may be produced by the process of signal generation (Section 2.3); indeed, tracks, like the ones we are working on, that encode statistical significance, such as the -log_{10}*p-value* of the signal compared to a control track, typically have a higher signal-to-noise ratio [3], possibly increasing the number of outliers in the tracks. So, to reduce their effect, we have decide to transform the data using three different functions: *logarithm* (log), *logarithm of the logarithm* ($\log\log$) and *inverse hyperbolic sine* ($\sinh^{-1}$) (Figure 3.10).

Once a prediction is computed, the result is properly reverted to the original scale for the evaluation task. Given a signal X_{ij} in the training set with i and j, referring to the cell and assay and, given x_{ijk} denoting the value of the signal at the k-th base pair, the transformations are applied

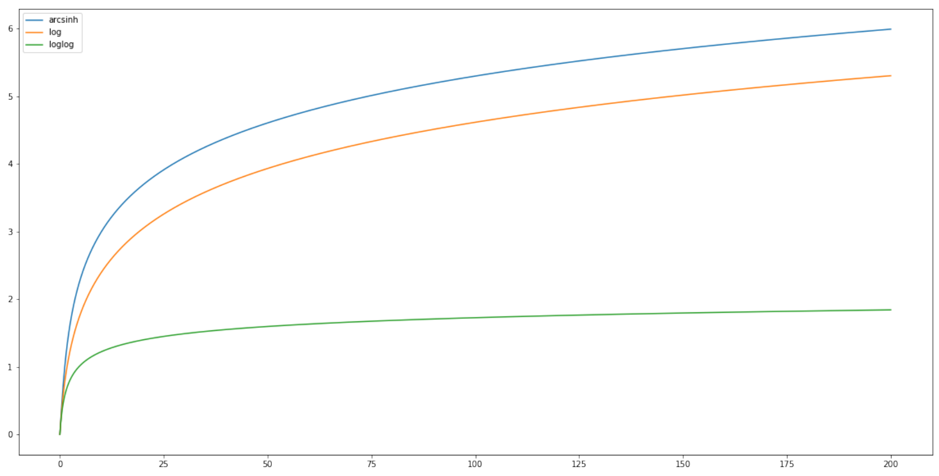


Figure 3.10: Transformation fuctions

to all signals in the dataset as follows:

- Log transformation:

$$\tilde{x}_{ijk} = \ln(x_{ijk} + 1) \tag{3.1}$$

- Loglog transformation:

$$\tilde{x}_{ijk} = \ln(\ln(x_{ijk} + 1) + 1) \tag{3.2}$$

- Inverse hyperbolic sine:

$$\tilde{x}_{ijk} = \sinh^{-1}(x_{ijk}) \tag{3.3}$$
$$\sinh^{-1}(x_{ijk}) = \ln(x_{ijk} + \sqrt{1 + x_{ijk}^2}) \tag{3.4}$$

The aim of the transformation is to reduce the large values without reducing too much the differences between small values and actually meaningful high values, otherwise, key information for the imputation would be lost in the process.

3.4 Evaluation metrics

Along with the *Imputation Challenge* are provided nine evaluation measures that will be used in order to score the ranking of the different submissions. The measures can be partitioned in two macro-groups: the ones related to the mean squared error (MSE) and ones related to measuring correlation between the ground truth and the prediction. The metrics are the following [15]:

- *MSE*: Standard mean square error between true signal and predicted signal.
- *MSE1obs*: Given the indexes of the top1% values of the true signal, mean squared error between the true signal and the predicted signal just considering the values associated to those positions.
- *MSE1imp*: Given the indexes of the top1% values of the predicted signal, mean squared error between the true signal and the predicted signal just considering the values associated to those positions.
- *GWcorr*: The Pearson correlation coefficient between the true signal and the predicted signal.
- *GWSpear*: The Spearman correlation between the predicted and true values.
- *MSEProm*: Similar to the global mean-squared error (MSE), but averaging is done only across genomic regions that are annotated as "promoters".
- *MSEGene*: MSE computed across regions annotated as "genes".
- *MSEEnh*: MSE computed across regions annotated as "enhancers."
- *MSEvar*: MSE weighted by cross-cell-type variance. Computing this measure involves computing, for an assay carried out in cell type x and assay type y, a vector of variance values across all assays of type y. The squared error between the predicted and true value at each genomic position is multiplied by this variance (normalized to sum to 1 across all bins) before being averaged across the genome.

3.4.1 Aggregating evaluation measures

Given the fact that the majority of the measures used in the challenge to score the imputed tracks are based on the MSE, if we are trying to optimize a prediction function and we need to chose an appropriate performance measure that can be aggregated on the validation set, the most obvious choice would be to optimize the average of the MSE between the predictions and the true signals. So, defined the predicted signal X_{ij} for a couple cell assay (i,j) and the corresponding true track Y_{ij}, the aggregated measure on the validation set would be:

$$\text{avgMSE}(\mathbf{X}) = \frac{1}{|\mathbf{X}|} \sum_{X_{ij} \in \mathbf{X}^{(predicted)}} \text{MSE}(X_{ij}, Y_{ij}) \tag{3.5}$$

Where $|\mathbf{X}|$ is the size of the validation dataset.

The problem with optimizing the average MSE value is that, as explained in the Section 3.2, some signals have low variance and may be relatively easy to predict, giving as a result low MSE values, while others have high variance and the active regions showcase huge values so it could be harder to predict with good accuracy. Thus, since we are averaging the MSE, the final result would be very sensitive to the latter signals, representing a problem because the overall performance of the teams in the competition is computed by averaging the quality (the ranking w.r.t. other teams) of the predictions on each predicted signal. Indeed, an high average MSE could hide the fact that we are achieving good results on the majority of the predicted samples because, on the few harder to predict signals, we are doing particularly bad, while, a relatively lower MSE, could introduce a bias towards certain algorithms even if we are not doing particularly well on each of the validation tracks. Thus, optimising performance only for hard to predict signals at the expense of the easy to predict ones could result in poor average rank performance. As a possible solution, instead of averaging MSE directly, we could optimise the Fraction of Variance Unexplained (FVU), which takes into account the variance of the signal being predicted, in order to normalise the MSE:

$$\mathrm{FVU}(X_{ij}, Y_{ij}) = \frac{\mathrm{MSE}(X_{ij}, Y_{ij})}{\sigma^2(X_{ij})} \tag{3.6}$$

$$\text{where} \;\; \sigma^2(X_{ij}) = \frac{1}{\mathcal{K}} \sum_k (x_{ijk} - \mu(X_{ij}))^2 \tag{3.7}$$

Alternatively, exploiting the results of the Avocado model, we can compute the *relative MSE* values with respect to the predictions of the baseline system, $Y_{ij}^{(base)}$:

$$\mathrm{relMSE}(X_{ij}, Y_{ij}; Y_{ij}^{(base)}) = \frac{\mathrm{MSE}(X_{ij}, Y_{ij})}{\mathrm{MSE}(X_{ij}, Y_{ij}^{(base)})} \tag{3.8}$$

Moreover, the *relative MSE* would be a much easier metric to interpret if we want to really understand how far behind/ahead we are from the baseline predictor. Note that, both FVU or relMSE are simply rescaled versions of MSE where the scaling coefficient depends on the signal but not the prediction (i.e. it is constant with respect to the model) and thus, optimising for these evaluation functions, is no more difficult than is optimising for MSE.
All the notation used in this chapter will be explained and formalized in the next one in order to guarantee uniformity across the explanation of the model.

Chapter 4

K-Nearest Neighbor model

Let's start describing the model by defining the notation that will be used in the following sections (Table 4.1):

Notation	**Description**
$\mathcal{I} = 51$	Number of cell types
i	$i - th$ cell type $\in \mathcal{I}$
$\mathcal{J} = 35$	Number of assays
j	$j - th$ assay type $\in \mathcal{J}$
$\mathcal{K} = 121241707$	Num. of genome locations (25 base-pair resolution)
$\mathcal{K}_m = 1000000$	One million genome locations (randomly sampled)
k	$k - th$ base pair $\in \mathcal{K}$ or $\mathcal{K}_m$
$\mathcal{L}$	Number of latent factors
l	$l - th$ latent factor $\in \mathcal{L}$
$X_{ij} = \{x_{ijk}\}_{k=1}^{\mathcal{K}}$	Signal for cell-type i, assay j
$x_{ijk} \in \mathbb{R}^{+} \cup \{0\}$	Value of signal for cell-type i, assay j at position k
$\mathbf{X}^{(train)} \subset \{X_{ij}\}_{i=1,j=1}^{\mathcal{I}\ \ \mathcal{J}}$	Training dataset (subset of all cell+assay pairs)
$Y_{ij}(\theta)$	Predicted signal for cell i assay j using parameters θ
$\mathcal{L}(X, Y)$	loss function $\mathcal{O}(\theta)$
objective function given the parameters θ	
$\mu(X_{ij})$	Mean of signal for cell i assay j
$\sigma(X_{ij})$	Standard deviation of signal X_{ij}
$Z_{ij} = \{z_{ijk}\}_{k=1}^{\mathcal{K}}$	Signal X_{ij} normalized with z-score
$z_{ijk} = \frac{x_{ijk} - \mu(X_{ij})}{\sigma(X_{ij})}$	Value normalized using z-transform
$\tilde{x}_{ijk} = \sinh^{-1}(x_{ijk})$	Value transformed using Inverse Hyperbolic Sine
$\rho(X_{ij}, X_{i'j'})$	Peason correl. between X_{ij} and $X_{i'j'}$
$\rho(i'i)$	Estimated Pearson correl. between cells i and i'
$\rho(jj')$	Estimated Pearson correl. between assays j and j'
$\vec{\rho}_w(X_{ij}, X_{i'j'})$	Pearson correl. between two signals at a given resolution w

Table 4.1: Notation

4.1 Similarities between samples

The preliminary step to the implementation of the KNN approach is to define the different similarities between all the signals in the training set. Other than computing the values necessary to describe the distance between the signals, defining how close a signal is to another can be very

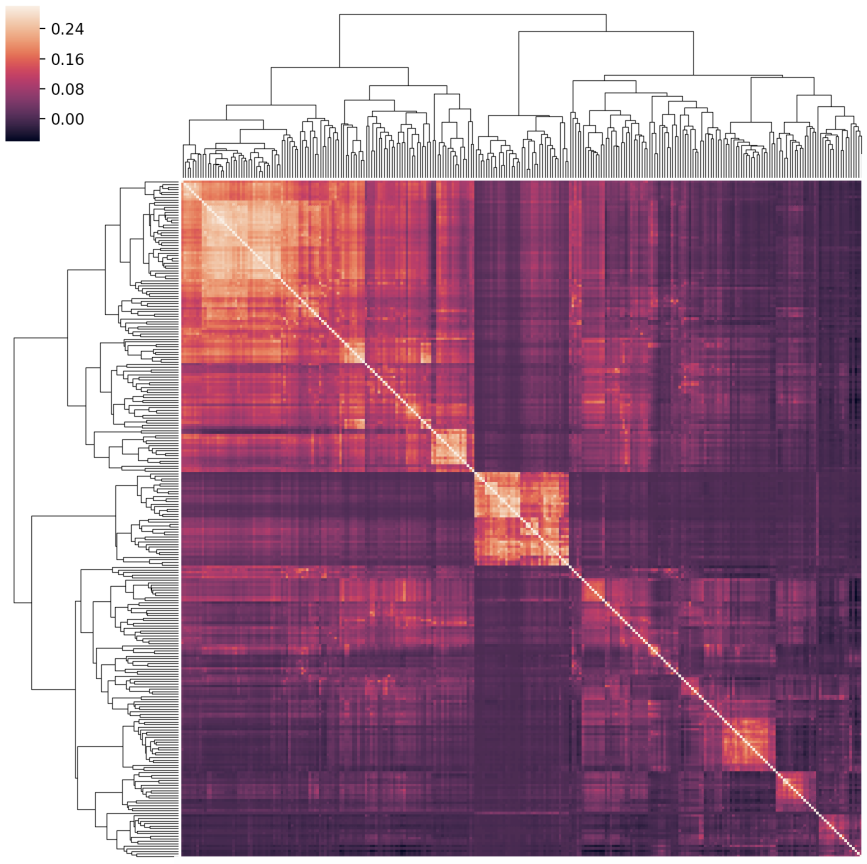
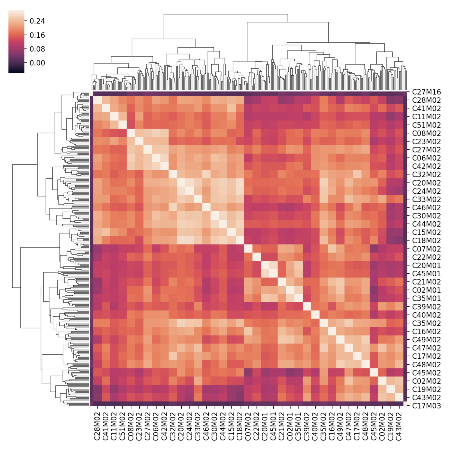


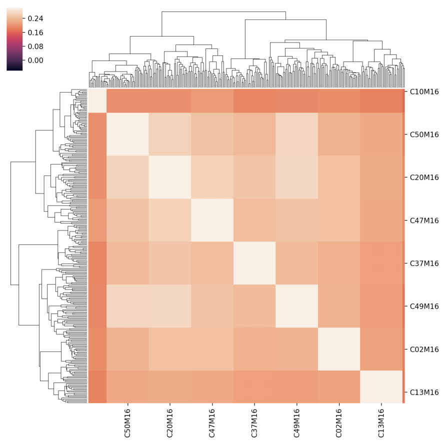
Figure 4.1: Heatmap of the similiarity matrix of the training samples

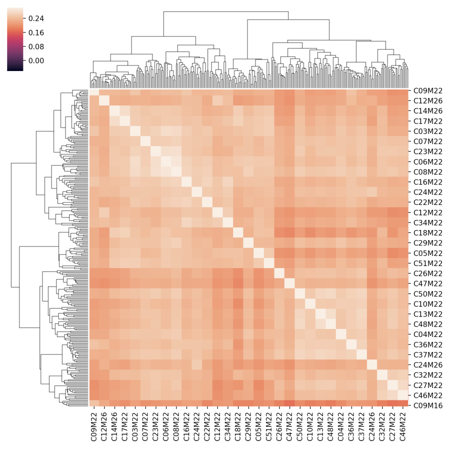
helpful in order to further understand the different relationships between assay-lines and cell-lines (how much we should rely on assay or cell dimensions in order to compute the prediction for a given target). First of all, let us denote the mean and (population) standard deviation for each signal as:

$$\mu(X_{ij}) = \frac{1}{\mathcal{K}} \sum_k x_{ijk} \quad \text{and} \quad \sigma(X_{ij}) = \sqrt{\frac{1}{\mathcal{K}} \sum_k (x_{ijk} - \mu(X_{ij}))^2} \tag{4.1}$$

The correlation between each signal will be computed with the Pearson correlation coefficient. Since the operation is invariant to scale, the output values should be fairly reliable estimators of which are the active regions along the genome and which ones are not. So, given two cell-assay couples (i, j) and (i', j') and the related signals X_{ij} and $X_{i'j'} \in X^{(train)}$, with $i \neq i' \in \mathcal{I}$ and $j \neq j' \in \mathcal{J}$ the similarity computation is formulated as


(a) Signals C●M02


(b) Signals C●M16


(c) Signals C●M22

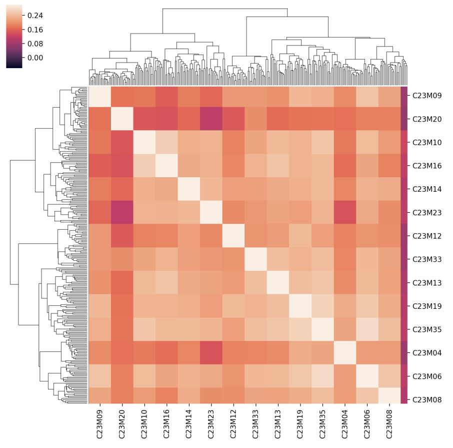

(d) Signals C23M●

Figure 4.2: Zooms on high-similarity quadrants of the heatmap (Figure 4.1)

follows

$$\rho(X_{ij}, X_{i'j'}) = \frac{\sum_k (x_{ijk} - \mu(X_{ij}))(x_{i'j'k} - \mu(X_{i'j'}))}{\sqrt{\sum_k (x_{ijk} - \mu(X_{ij}))^2}\sqrt{\sum_k (x_{i'j'k} - \mu(X_{i'j'}))^2}} \tag{4.2}$$

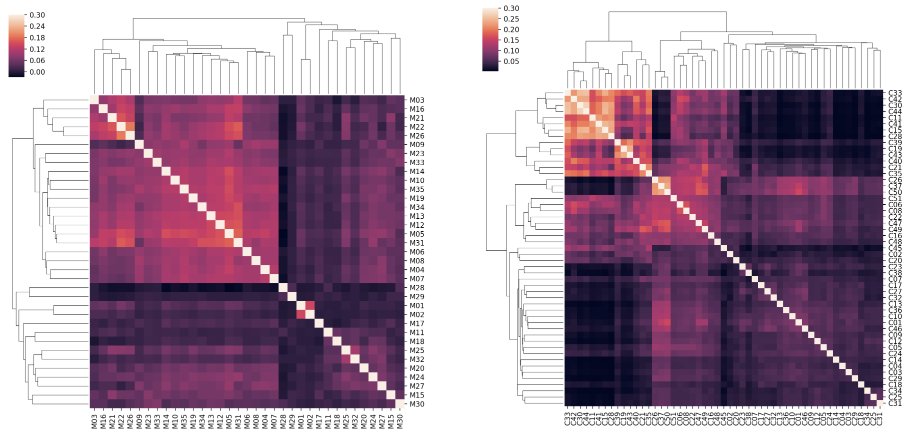
$$= \frac{1}{\mathcal{K}} \sum_k z_{ijk} z_{i'j'k} \tag{4.3}$$

where z_{ijk} denotes the normalized signal value for the base pair k:

$$z_{ijk} = \frac{x_{ijk} - \mu(X_{ij})}{\sigma(X_{ij})} \tag{4.4}$$

The final result can be visualized through a clustermap of dimension 267×267 (Figure 4.1) where can be clearly seen how there are group of signals similar to each other and, at the same time, very different from the rest of the training set samples. What can give better information can be zooming into the heatmap and analyze which are the signals belonging to the different clusters identified. For this purpose, as an example, looking at Figures 4.2 it is possible to understand that the signals belonging to the same cluster most likely refer to a certain cell-line or a certain assay-line (for the majority of the clusters), showcasing how an approach like the KNN regression, that aims at exploiting signals on the same dimension to predict a target, can achieve good performances. On top of that, the fact that the majority of the clusters groups are due to similar signals belonging to the same assay but different cells supports the results highlighted by the analysis of the differences between signals distributions reported in the section 3.2.1.

4.1.1 Aggregating similarities



(a) Assay by assay similarities (b) Cell by cell similarities

Figure 4.3: Estimated similarity values

In order to exploit the already presented similarities as distance measures between the training set signals and blind instances, is necessary to estimate a value able to define how close a signal $X_{ij} \in X^{(train)}$ will be to the target $Y_{i'j'}$. Since the predictions in the KNN will be performed exploiting signals along the same dimension (cell and assay) the estimated value will correspond to the similarities between cells and assays, resulting in two different similarity matrices, respectively of dimensions $\mathcal{I} \times \mathcal{I}$ and $\mathcal{J} \times \mathcal{J}$ (Figure 4.3). Practically, an efficient way to compute the similarity between two cell-lines i and i', is to simply average all the Pearson correlation $\rho(X_{ij}, X_{i'j})$ values for all the assays j such that $X_{ij}, X_{i'j} \in X^{(train)}$:

$$\rho(i, i') = \frac{\sum_{j: X_{ij}, X_{i'j} \in \mathbf{X}^{(train)}} \rho(X_{ij}, X_{i'j})}{\sum_{j: X_{ij}, X_{i'j} \in \mathbf{X}^{(train)}} 1} \tag{4.5}$$

Analogously, it can be computed the similarity between assays j and j':

$$\rho(j, j') = \frac{\sum_{i: X_{ij}, X_{ij'} \in \mathbf{X}^{(train)}} \rho(X_{ij}, X_{ij'})}{\sum_{j: X_{ij}, X_{ij'} \in \mathbf{X}^{(train)}} 1} \tag{4.6}$$

4.2 KNN regression

The first approach of the thesis consists in performing a type of k-Nearest Neighbour based Regression over the space of cell-types and assays. The main idea is to exploit the estimated cell-by-cell and assay-by-assay similarities (see Chapter 3) between the pairs of signals (in the training data) and use those to construct the unknown signal via a weighted average of the known signals.
It is a simple approach but fits the structure of the data and, considering the high sparsity of the tensor which comes with a low total number of signals, it is possible that an instance based method, using directly the training data as the knowledge of the problem, can outperform the methods based on learning the model parameters from the training set. The k-Nearest neighbour regression will be performed approaching the data from two of the three dimensions of the tensor, the cell dimension and the assay dimension. Indeed, trying to exploit the prior knowledge about the similarities between the same assay applied on different cells and the other way around can be an useful starting point able to deliver decent performances. More specifically, for the weighted nearest neighbour regressor we linearly combine various training signals together, weighting each signal by the estimated similarity to the target signal coming from the cell-by-cell or assay-by-assay similarity matrix. The predictions make use of all signals $X_{\bullet j}$ that involve the target assay j or all the signals $X_{i\bullet}$ that involve the target cell i, weighting the contribution of each signal by the similarity of the corresponding cell/assay-line to the cell/assay-line that needs to be predicted.

4.2.1 Assay based predictions

The assay based predictions exploits the knowledge about all the signals across the same assay-line in order to predict a target track that as well belongs to that specific assay-line.
Given a target signal Y_{ij} with $i \in \mathcal{I}$ and $j \in \mathcal{J}$ the prediction is performed through the weighted average of all the signals $X_{i'j} \in X^{(train)}$.

The weighted average is performed exploiting all the estimated similarities $\rho_{i'i}$ between the target Y_{ij} and the existent signals $X_{i'j}$ in the training set:

$$Y_{ij}^{(assay)} = \frac{\sum_{i':X_{i'j}\in\mathbf{X}^{(train)}} \rho_{i'i} X_{i'j}}{\sum_{i':X_{i'j}\in\mathbf{X}^{(train)}} \rho_{i'i}} \tag{4.7}$$

4.2.2 Cell based predictions

The cell-based prediction is analogous in that and, given a target signal Y_{ij}, it combines all the training signals $X_{ij'}$ that have the same cell-type as the target signal in order to build the prediction. It uses in the same way the estimated similarities $\rho_{jj'}$ between each training signal's assay and the target assay to weight the contribution of each signal:

$$Y_{ij}^{(cell)} = \frac{\sum_{j':X_{ij'}\in\mathbf{X}^{(train)}} sim(j,j') X_{ij'}}{\sum_{j':X_{ij'}\in\mathbf{X}^{(train)}} sim(j,j')} \tag{4.8}$$

In both the above assay and cell based predictions the approach has been formalized considering the weighted average between all the samples, but it is possible that considering all the available information across a given cell-line i or assay line j could introduce noise in the final prediction due to low-similarity value between a considered training sample and the target signal. Indeed, the predictions can be restricted averaging only the top-k most similar signals by replacing $\mathbf{X}^{(train)}$ with an appropriately chosen subset. See Section 4.2.6 for a more detailed explanation about the different modalities that can be used in order to select the k parameter.

4.2.3 Rescaling Assay and Cell-based predictions

As seen from the visualizations in Chapter 3, the nature itself of some cells or assays can generate a consistent difference in scale between the different signals, so, there is no guarantee that tracks for different cell types or more importantly for different assays, will be on the same scale as one another (Figure 4.4). Since the most important task is, given a couple cell-assay, to identify which are the active regions of the genome we used a scale-free method (invariant to rescaling), Pearson correlation coefficient in order to estimate the different similarities. However, when generating a prediction, it is important that the scale of the predicted values agrees with the one of the true values for two different reasons: identifying which gene is active may not be meaningful enough if the scale doesn't reflect correctly the confidence with which there will be biochemical activity along that section of the genome, plus, since the main measure used to evaluate the prediction is the mean squared error, differences in the scale can penalize significantly the final evaluation both in the challenge and with reference

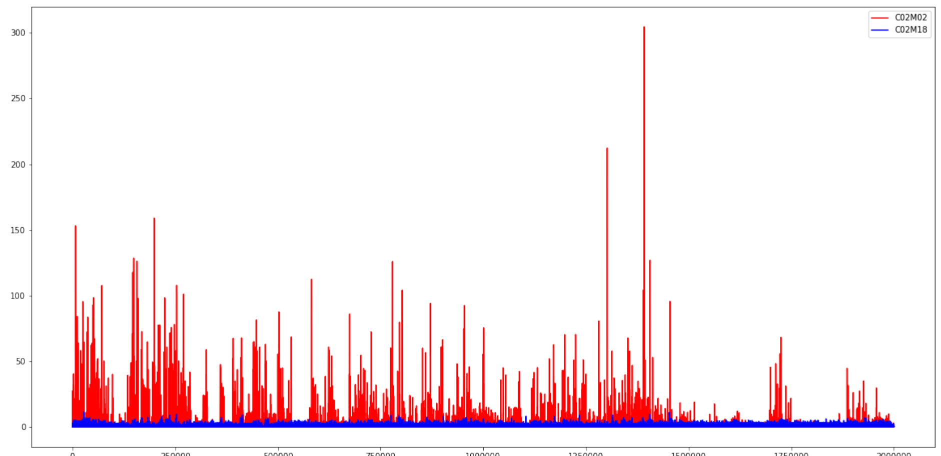


Figure 4.4: Signals C02M02,C02M18 chromosome 1, bp[6:8]1e6

to the Avocado baseline. So, since it may be the case that the linear combination of similar signals computed above does not have the correct scale we tried to improve the scale of the predicted signal by modifying the prediction rules from above to include an estimated scale factor as follows. For the assay/cell-based prediction we would then have:

$$Y_{ij}^{(assay)} = \frac{\hat{\sigma}_{ij}^{(assay)} \sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} \rho_{i'i} \frac{X_{i'j}}{\sigma(X_{i'j})}}{\sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} \rho_{i'i}} \tag{4.9}$$

$$Y_{ij}^{(cell)} = \frac{\hat{\sigma}_{ij}^{(cell)} \sum_{j':X_{ij'} \in \mathbf{X}^{(train)}} \rho_{jj'} \frac{X_{ij'}}{\sigma(X_{ij'})}}{\sum_{i':X_{ij'} \in \mathbf{X}^{(train)}} \rho_{jj'}} \tag{4.10}$$

If the signals when averaged are normalized by their standard deviation, predicting the scale of the target signal can be done using an *unweighted* average of the standard deviations across the training signals for that cell-type:

$$\hat{\sigma}_{ij}^{(cell)} = \frac{\sum_{j':X_{ij'} \in \mathbf{X}^{(train)}} \sigma(X_{ij'})}{\sum_{j':X_{ij'} \in \mathbf{X}^{(train)}} 1} \tag{4.11}$$

Or likewise, an unweighted average across the signals for the same assay:

$$\hat{\sigma}_{ij}^{(assay)} = \frac{\sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} \sigma(X_{i'j})}{\sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} 1} \tag{4.12}$$

In this case, in order to normalize the signals, we are just using the standard deviation; it might be possible that using the z-score transformation the performance can improve. The z-score transformation Z_{ij} of the corresponding signal X_{ij} can be obtained by normalizing each value of the vector like follows:

$$z_{ijk} = \frac{x_{ijk} - \mu(X_{ij})}{\sigma(X_{ij})} \tag{4.13}$$

Once the z-score transformation is performed, in order to compute the prediction, will also be necessary to estimate the average value $\hat{\mu}_{ij}$ of the target signal we are going to predict. In order to do this the simplest possibility could be to just average all the averages of the signals necessary to perform the target assay or cell based prediction:

$$\hat{\mu}_{ij}^{(cell)} = \frac{\sum_{j':X_{ij'} \in \mathbf{X}^{(train)}} \mu(X_{ij'})}{\sum_{j':X_{ij'} \in \mathbf{X}^{(train)}} 1} \tag{4.14}$$

$$\hat{\mu}_{ij}^{(assay)} = \frac{\sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} \mu(X_{i'j})}{\sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} 1} \tag{4.15}$$

The final prediction could the be computed like this:

$$Y_{ij}^{(assay)} = \frac{\hat{\sigma}_{ij}^{(assay)} \sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} \rho_{i'i} Z_{i'j}}{\sum_{i':X_{i'j} \in \mathbf{X}^{(train)}} \rho_{i'i}} + \hat{\mu}_{ij} \tag{4.16}$$

The same formulation goes for the cell based prediction.
Besides the simple average prediction, in order to improve the quality of the imputed track, it could be possible to learn a function to predict the scale for the feature vector: (i, j).

4.2.4 Predicting at different resolutions

Up until this point we have assumed that the similarity between signals is computed globally over the entire signal. Given that the human genome is composed of 23 chromosomes which function somewhat independently of one another, and, moreover, that signals along each chromosome are correlated in space, it makes sense to investigate similarity measures that are local to parts of the signal. The process will be identical to the computation of the global similarity but will be repeated across the genome several times depending on the selected resolution. So, the first step is to define a partition of the genome that generates several sections, each one corresponding to one of the three different resolutions/length segments $w \in \{10^6, 10^5, 10^4\}$ adopted, and then apply it to each sample track in the training set $X^{(train)}$. The resolution we are working on is always considering the genome at 25 base pairs, so, considering its full length the actual dimension of each segment would be: $w \in \{25 * 10^6, 25 * 10^5, 25 * 10^4\}$.

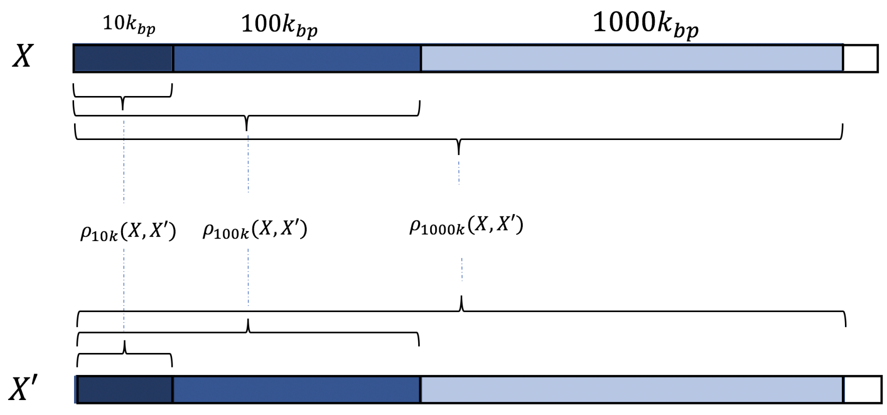


Figure 4.5: Pearson correlation coeff. at different resolutions

Anyway, given the partition, it is now possible to investigate the similarity between the different tracks for a certain section in the genome and so, for each of the segments, to compute separately the Pearson correlation coefficient. Considering two signals X_{ij} and $X_{i'j'}$ the similarity between the two will now be a vector of dimension $\frac{\mathcal{K}}{w}$ equal to:

$$\vec{\rho}_w(X_{ij}, X_{i'j'}) = \{\rho(\{x_{ij1}, ..., x_{ijw}\}, \{x_{i'j'1}, ..., x_{i'j'w}\}), ...\} \tag{4.17}$$

The final result of the computation of all the similarities between all the training samples is a matrix of dimension $|X^{(train)}| \times |X^{(train)}| \times \frac{\mathcal{K}}{w}$. The aggregation process performed in order to estimate the similarities between different cell-lines and different assay-lines is carried on by averaging independently for each section of the partition, following the same Equations 4.5 and 4.6 producing ultimately two matrices with dimension $\mathcal{I} \times \mathcal{I} \times \frac{\mathcal{K}_{25}}{w}$. The same goes for the assays $\mathcal{J}$:

$$\vec{\rho}_w(i, i') = \{\rho_w(i, i')_1, ..., \rho_w(i, i')_{\lceil \frac{\mathcal{K}_{25}}{w} \rceil}\} \tag{4.18}$$

$$\vec{\rho}_w(j, j') = \{\rho_w(j, j')_1, ..., \rho_w(j, j')_{\lceil \frac{\mathcal{K}_{25}}{w} \rceil}\} \tag{4.19}$$

The predictions are always performed across the assay or cell dimensions exploiting the previous Equations (4.7), (4.8). It is possible to improve the performance of the prediction through out the combination of the local similarities with the global similarities. In fact, since local and global information can be complementary (and local correlation values might exhibit high variance), it might make sense then to smooth each local estimates with the global correlation measure formulating the correlation computation like follows:

$$\vec{\rho}_w(X_{ij}, X_{i'j'})'_1 = \alpha \vec{\rho}_w(X_{ij}, X_{i'j'})_1 + (1 - \alpha)\rho(X_{ij}, X_{i'j'}) \tag{4.20}$$

Dot product similarity

An issue that arises when computing local correlation estimates is that over relatively short windows (of size say $w = 10^4$), the signal can often be flat, resulting in a very low or zero value for the standard deviation. Whenever this occurs the correlation values for the section can become either very large or undefined. To prevent this from occurring, a possible solution is to modify the Equation 4.17 to use global estimates for the mean and standard deviation when calculating correlation in each window. This can be done very easily by first normalizing the entire signal using the z-transform and then simply computing the dot-product for each section of the normalised signal:

$$\vec{\rho}_w(X_{ij}, X_{i'j'}) = \{\sum_{k=1}^{w} z_{ijk} z_{i'j'k}, \sum_{k=w+1}^{2w} z_{ijk} z_{i'j'k}, ...\} \tag{4.21}$$

4.2.5 Merging the predictions

The next step in order to increase the prediction performances is to use at the same time the information coming from both cells and assays, trying to aggregate the track imputed by the assay and cell based predictions.
We tried to approach the merging task from two different perspectives: it is possible to assume that the similarities in both assay and cell directions are comparable or that the merge can be computed just from a linear combination between the assay and cell predictions. The former tries to normalize concurrently the similarities between cells and assays as follows:

$$Y_{ij}^{(assay+cell)} = \frac{\sum_{i':X_{i'j} \in \mathbf{X}} sim(i,i') X_{i'j} + \sum_{j':X_{ij'} \in \mathbf{X}} sim(j,j') X_{ij'}}{\sum_{i':X_{i'j} \in \mathbf{X}} sim(i,i') + \sum_{j':X_{ij'} \in \mathbf{X}} sim(j,j')} \tag{4.22}$$

The latter aims at aggregating assay and cell based predictions by finding a fixed coefficient *alpha*, combining them as follows:

$$Y_{ij}^{(aggregate)} = \alpha Y_{ij}^{(assay)} + (1-\alpha) Y_{ij}^{(cell)} \tag{4.23}$$

Adapting the α coefficient

Instead of using a fixed coefficient α for combining the assay-based and cell-based predictions in Equation 4.23, it could be possible to adapt the parameter for each target signal $Y_{i,j}$, exploiting the knowledge extracted from the training set with the estimated similarities cell-by-cell and assay-by-assay. Indeed, we could allow the coefficient to vary based on the amount of similar signals we get in each direction i and j for predicting Y_{ij}:

$$\alpha_{ij} = \frac{\sum_{i':X_{i'j} \in X^{(train)}} \rho(i,i')}{\sum_{i':X_{i'j} \in X^{(train}} \rho(i,i') + \sum_{j':X_{ij'} \in X^{(train)}} \rho(j,j')} \tag{4.24}$$

The idea behind the formula is to give more weight to the assay or cell prediction according to two different factors: (i) if the summation of the similarities between the target and train cell or assay is greater in on one of the two dimensions, it will be more likely that the axis with the "more similar" signals will deliver better predictions (ii) if the amount of signals is greater in one of the two dimensions (e.g. there are cases where there is just one signal for a given cell-line) probably that dimension will be able to capture a larger amount of information necessary to predict the target signal.
A problem that can arise is that, trying to predict a target signal Y_{ij}, the amount of signals $X_{\bullet j} \in X^{(train)}$ is much larger than the one of the signals $X_{i\bullet}$ and at the same time the majority of the similarities $\rho(i, i')$ has a low Pearson correlation coefficient, ending up valuing more a prediction that combines a larger number of tracks but that are not that similar to the target we need to impute.
It is then possible to define a fixed K_j and K_i subsets of the signals, available in the training data for the same assay j and cell-line i of the signal that we are predicting with Equation (4.23), redefining the way α_{ij} is computed:

$$\alpha_{ij} = \frac{\sum_{i':X_{i'j} \in K_j} \rho(i, i')}{\sum_{i':X_{i'j} \in K_j} \rho(i, i') + \sum_{j':X_{ij'} \in K_i} \rho(j, j')} \tag{4.25}$$

Then, considering Y_{ij} as the target signal, the sets K_j and K_i can be defined as follows:

- *threshold*: include only signals $X_{i'j}$ or $X_{ij'}$ s.t. the $\rho(i'i)$ or $\rho(jj')$ similarity is higher than a threshold ϵ:

$$\begin{aligned} K_j &= \{X_{i'j} \in \mathbf{X} | \rho(i, i') \geq \epsilon\} \\ K_i &= \{X_{ij'} \in \mathbf{X} | \rho(j, j') \geq \epsilon\} \end{aligned} \tag{4.26}$$

- *top-k*: include only signals $X_{i'j}$ or $X_{ij'}$ s.t. the $\rho(i'i)$ or $\rho(jj')$ are the *topK* similarities:

$$\begin{aligned} K_j &= \{X_{i'j} \in \mathbf{X} | \rho(i, i') \geq \epsilon_{ijk}^{(i)}\} \\ K_i &= \{X_{ij'} \in \mathbf{X} | \rho(j, j') \geq \epsilon_{ijk}^{(j)}\} \end{aligned} \tag{4.27}$$

where $\epsilon_{ij}^{(i)} = sort(\{\rho(i, i') | X_{i'j} \in \mathbf{X}\})$ is the sorted list of cell-line similarity values (from highest to lowest) for which the signal $X_{i'j}$ is present in the training data, and $\epsilon_{ijk}^{(i)}$ then denotes the k^{th} highest value in the list.

- *top-k%*: same as *top-k* above except that rather than restricting to the top k values, we restrict to the top $k\%$ of the signals available for that assay / cell-line.

4.2.6 Top-k Assay and Cell based predictions

As the last aspect of the KNN regressor approach, we consider to select the k-nearest neighbors when performing the Assay and Cell based predictions of equations (4.7), (4.8). Just considering as an example the Assay based prediction, instead of imputing the target track Y_{ij} considering all the signals $X_{i'j} \in X^{(train)}$, we consider the list $\epsilon_{ij}^{(i)} = sort(\{\rho(i,i')|X_{i'j} \in \mathbf{X}\})$, which is the sorted list of cell-line similarity values (from highest to lowest), we restrict it to the top $\bullet\%$ higher values, and we select those couples $(\rho(i'i), X_{i'j})$ to predict the target instance. The same process can be applied for the Cell based predictions. It is also important to say that the $\bullet\%$ of tracks used is fixed independently from which Y_{ij} track we are going to predict. The best percentage will be selected based of the results obtained on the validation set as will be shown in the Chapter 6.

Chapter 5

Factorization models

5.1 Factorization models

In this section I will describe how we approached the imputation task through the other two algorithms mentioned in the background chapter (2): matrix factorization and tensor factorization. These kind of models fit the way the dataset is structured and they can, in fact, be considered one of the main candidates to chose when the the data can be represented as a three dimensional tensor and we want to predict the missing values. Indeed, two of the presented state of art works employ the PARAFAC tensor decomposition as the core of their algorithms.
Given the fact that previous methods were able to perform with success the imputation task, we also decided to implement the same idea of decomposing and characterize the information of the data, trying to improve the current state of the algorithms. Thus, we introduced both a novel approach to the subject with a base-pair-wise matrix factorization and new ideas about the structure of the PARAFAC tensor decomposition used by the baselines.

5.1.1 Objective function

First of all, since the models will be trained through a gradient descent technique, we need to define an objective function that we want to minimize through the learning of the parameters.
Given the fact that the evaluation in the *Imputation Challenge* is carried on by mainly exploiting metric based on the mean square error, the choice is pretty straightforward. Indeed, defined with θ the parameters of the model with respect to the training instances $\mathbf{X}^{(train)}$, the loss function $\mathcal{O}$ can be structured as the summation of the different $\text{MSE}(X_{ij}, Y_{ij})$ between

the target tracks of the training set and the respective predicted signals:

$$\begin{aligned} \mathcal{O}(\theta; \mathbf{X}) &= \sum_{(i,j)\in\mathbf{X}} \mathcal{L}(X_{ij}, Y_{ij}(\theta)) + \mathcal{R}(\theta) & (5.1) \\ &= \sum_{(i,j)\in\mathbf{X}} \text{MSE}(X_{ij}, Y_{ij}(\theta)) + \mathcal{R}(\theta) & (5.2) \end{aligned}$$

Here, $\mathcal{R}$ denotes the regularization function. The purpose of adding regularization here is to prevent overfitting of the learnt parameters. More specifically, the regularization terms that will be adopted in the respective methodologies are the Lasso regularisation term, Ridge and the Elastic net regularization term. Across the training of the model we will not compute the gradients over the overall objective function, instead, with a stochastic gradient descent (SGD) approach we will try to update the parameters based on one signal at a time; so the minimization of the $\mathcal{O}(\theta; \mathbf{X})$ objective function will be achieved through the minimization of the single components of the summations. Thus, the objective function can be rewritten in an more intuitive way as a summation over instance-wise objective functions $\mathcal{O}(\theta; X_{ij})$ for each signal X_{ij} as follows:

$$\mathcal{O}(\theta; \mathbf{X}) = \sum_{(i,j)\in\mathbf{X}} \mathcal{O}(\theta; X_{ij}) \quad (5.3)$$

$$where \;\; \mathcal{O}(\theta; X_{ij}) = \text{MSE}(X_{ij}, Y_{ij}(\theta)) + \frac{1}{|\mathbf{X}|}\mathcal{R}(\theta) \quad (5.4)$$

Here, $|\mathbf{X}|$ denotes the number of training signals, i.e., distinct pairs (i, j) in the training data.

5.2 Additive model

A preliminary step before matrix and tensor factorization is to try and define an additive model, which considers just the biases of the matrix factorization. The idea is similar to the KNN regressor one; indeed, the aim is to combine the information coming from both assay and cell dimension in order to predict a target signal Y_{ij}. In this case, instead of estimating the similarities between cell and assay-lines and combine the signals, we try to learn, throughout SGD, the average information that characterizes each assay and cell across the genome.

Thus, for each cell i and each assay j we will have two vectors A_i and B_j that, summed together, represent the prediction Y_{ij} (5.1):

$$Y_{ij} = A_i + B_j \quad (5.5)$$

Where $A_i \in \mathbb{R}^{\mathcal{K}}$ and $B_j \in \mathbb{R}^{\mathcal{K}}$ are representative signals/biases for cell-line i and assay j, respectively.

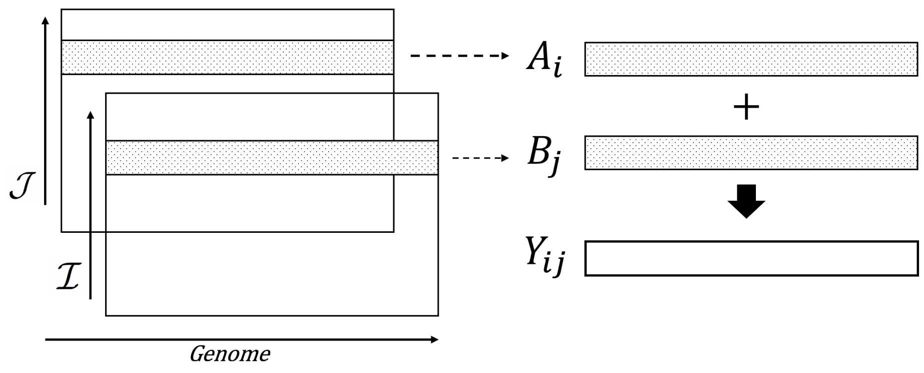


Figure 5.1: Additive model

5.2.1 Additive model parameter estimation

As already mentioned above, the objective function is based on the MSE metric. Thus, considering the linear prediction function $Y_{ij} = A_i + B_j$ for a particular signal X_{ij} the metric can be defined as:

$$\text{MSE}(X_{ij}, Y_{ij}) = \frac{1}{\mathcal{K}} \sum_{l} (x_{ijk} - y_{ijk})^2 = \frac{1}{\mathcal{K}} \sum_{l} (x_{ijk} - (a_{ik} + b_{jk}))^2 \quad (5.6)$$

Where a_{ik} refers to the k-th base pair value of the genome in vector A_i and so b_{jk} for vector B_{jk}. The overall objective function and the objective function for each sample of the training set will be then defined according to equations (5.4) and (5.3). The regularization function applied to the parameters of the model could be either L_2 (Gaussian) or L_1 (Laplace, sparsity-inducing) norms. The latter may be more useful once the number of parameters increases with matrix factorization or tensor factorization (always depending on the number of chosen latent factors). So, we can formalize two different regularization terms for the additive model:

$$\mathcal{R}(\theta) = L_2(\theta) = \lambda(\sum_{i}\sum_{k} a_{ik}^2 + \sum_{j}\sum_{k} b_{jk}^2) \quad (5.7)$$

$$\mathcal{R}(\theta) = L_1(\theta) = \lambda(\sum_{i}\sum_{k} |a_{ik}| + \sum_{j}\sum_{k} |b_{jk}|) \quad (5.8)$$

As already mentioned, the gradients will be computed considering the single instances of the training set exploiting the objective function $\mathcal{O}(\theta; X_{ij})$. For the additive model, where the biases are the parameters, for each training sample X_{ij}, we will compute the gradients and update the corresponding A_i and B_j vectors.
Base pair-wise, with respect to the parameter a_{ijk} and assuming the first

(L_2) regularisation term, the derivative can be formalized as follows:

$$\frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial a_{ik}} = -\frac{2}{\mathcal{K}}(x_{ijk} - (a_{ik} + b_{jk})) + 2\frac{\lambda}{|\mathbf{X}|}a_{ik} \tag{5.9}$$

$$= -\frac{2}{\mathcal{K}}e_{ijk} + 2\frac{\lambda}{|\mathbf{X}|}a_{ik} \tag{5.10}$$

where, $e_{ijk} = (x_{ijk} - (a_{ik} + b_{jk}))$ denotes the residual error in the prediction at position k, and $|X|$ corresponds to the cardinality of the set of training instances. For the sparse regularisation term (L_1) the partial derivative can be formulated as follows:

$$\frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial a_{ik}} = -\frac{2}{\mathcal{K}}e_{ijk} + \frac{\lambda}{|\mathbf{X}|} * sgn(a_{ik}) \tag{5.11}$$

Where $sgn()$ denotes the sign function with values $\{-1, 0, 1\}$ depending on whether the argument is less-than, equal-to or greater-than zero.

Analogous partial derivatives can be derived with respect to b_{jk} for the two different regularisation terms:

$$\frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial b_{jk}} = -\frac{2}{\mathcal{K}}e_{ijk} + 2\frac{\lambda}{|\mathbf{X}|}b_{jk} \tag{5.12}$$

$$\frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial b_{jk}} = -\frac{2}{\mathcal{K}}e_{ijk} + \frac{\lambda}{|\mathbf{X}|} * sgn(b_{jk}) \tag{5.13}$$

In order to estimate the parameters A_i and B_j after the their initialization, which will be explained in the experiment section (Chapter 6), we update the values by stochastic gradient descent, iterating over the signals $X_{ij} \in X^{(train)}$ and performing a gradient update until the model converges:

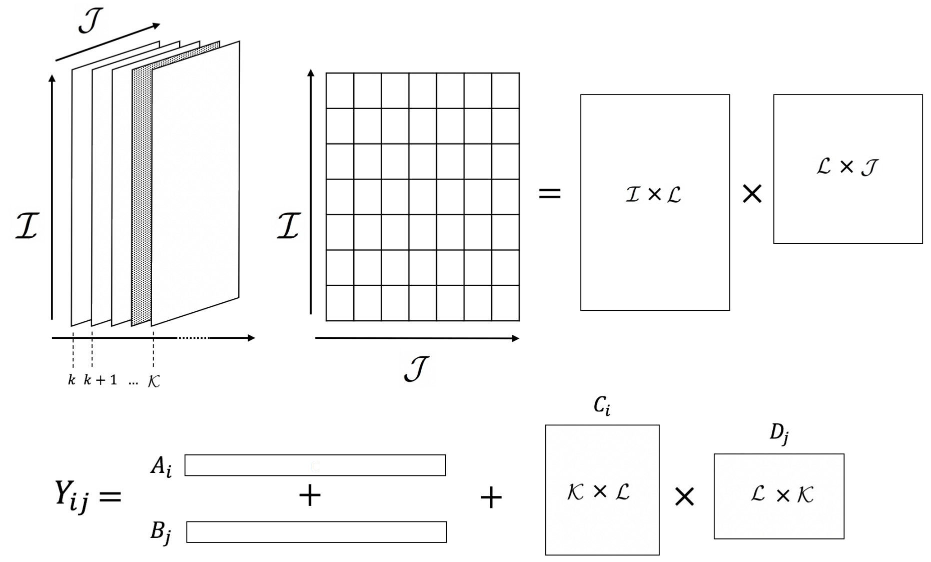
$$a_{ik}^{(t+1)} = a_{ik}^{(t)} - \eta\frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial a_{ik}} \tag{5.14}$$

$$b_{jk}^{(t+1)} = b_{jk}^{(t)} - \eta\frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial b_{jk}} \tag{5.15}$$

Here η is the step-size/learning rate that needs to be set empirically to guarantee convergence.

5.3 Matrix factorization

Next model, which is an extension of the additive one, is the matrix factorization approach.
Most of the state-of-the-art models presented in chapter 2 consider the base pairs of the genome as a valuable dimension that embeds part of the information brought by the different signal tracks of the three dimensional



$\mathcal{J}$
$\mathcal{I}$
k $k+1$ ... $\mathcal{K}$
$\mathcal{I}$
$\mathcal{J}$
=
$\mathcal{I} \times \mathcal{L}$
×
$\mathcal{L} \times \mathcal{J}$
$Y_{ij} =$
A_i
+
B_j
+
C_i
$\mathcal{K} \times \mathcal{L}$
×
D_j
$\mathcal{L} \times \mathcal{K}$

Figure 5.2: Matrix factorization model

tensor. In this case, we wanted to try to understand, as already introduced in 2.6, weather or not all that information necessary to impute the signals could instead be derived from just the characterization of the different cells and assays across the entire genome $\mathcal{K}$. A more intuitive way, useful to comprehend how the method works, is to imagine the tensor not as a single unit but as a series of $\mathcal{K}$ independent matrices $\mathcal{I} \times \mathcal{J}$ (Figure 5.2). We aim, in fact, at summarizing the information kept inside each base pair value for each signal into the assay and cell dimensions and, to do so, each one the K matrices is decomposed through out matrix factorization. Thus, given the target signal Y_{ij}, the prediction of the extended additive model that includes also the interaction between cell-line i and the assay j, can be formalized as follows:

$$Y_{ij} = A_i + B_j + C_i^T D_j = A_i + B_j + \sum_{l=1}^{\mathcal{L}} C_{il} D_{jl} \tag{5.16}$$

Here, $\mathcal{L}$ denotes the number of latent dimensions (probably only 2 given the amount of training data available) and $C_i \in \mathbb{R}^{\mathcal{L}\mathcal{K}}$ and $D_j \in \mathbb{R}^{\mathcal{L}\mathcal{K}}$ are factor matrices for the cell-line i and assay j.

5.3.1 Matrix factorization parameter estimation

Also in this case the objective function is based on MSE and follows the equations (5.3), (5.4). Thus, the MSE for the true signal X_{ij} and the predicted signal Y_{ij} can be written as follows:

$$\text{MSE}(X_{ij}, Y_{ij}) = \frac{1}{\mathcal{K}} \sum_{l} (x_{ijk} - (a_{ik} + b_{jk} + \sum_{l=1}^{\mathcal{L}} c_{ilk} d_{jlk}))^2 \quad (5.17)$$

Where c_{ilk} and d_{jlk} correspond, respectively, to the (l, k) elements of the matrices C_i and D_j relative to the current couple of cell and assay.
For the matrix factorization model, given the additional parameters, the L_2/L_1 regularization terms can be expressed as follows:

$$L_2(\theta) = \lambda \sum_{k} (\sum_{i} a_{ik}^2 + \sum_{j} b_{jk}^2 + \sum_{l} (\sum_{i} c_{ilk}^2 + \sum_{j} d_{jlk}^2)) \quad (5.18)$$

$$L_1(\theta) = \lambda \sum_{k} (\sum_{i} |a_{ik}| + \sum_{j} |b_{jk}| + \sum_{l} (\sum_{i} |c_{ilk}| + \sum_{j} |d_{jlk}|)) \quad (5.19)$$

The partial derivative with respect to the additional parameter c_{ilk}, depending on whether we are using L_2 or L_1 regularization, is expressed as follows

$$\frac{\partial O(\theta; X_{ij})}{\partial c_{ilk}} = -\frac{2}{\mathcal{K}} e_{ijk} d_{jlk} + 2 \frac{\lambda}{|\mathbf{X}|} c_{ilk} \quad (5.20)$$

$$\frac{\partial O(\theta; X_{ij})}{\partial c_{ilk}} = -\frac{2}{\mathcal{K}} e_{ijk} d_{jlk} + \frac{\lambda}{|\mathbf{X}|} * sgn(c_{ilk}) \quad (5.21)$$

where e_{ijk} denotes the residual error in the predicted value for position k in signal X_{ij}:

$$e_{ijk} = x_{ijk} - y_{ijk} = x_{ijk} - (a_{ik} + b_{jk} + \sum_{l=1}^{\mathcal{L}} c_{ilk} d_{jlk}) \quad (5.22)$$

Analogous partial derivatives can be formalized with respect to d_{jlk} for the two different regularization terms:

$$\frac{\partial O(\theta; X_{ij})}{\partial d_{jlk}} = -\frac{2}{\mathcal{K}} e_{ijk} c_{ilk} + 2 \frac{\lambda}{|\mathbf{X}|} d_{jlk} \quad (5.23)$$

$$\frac{\partial O(\theta; X_{ij})}{\partial d_{jlk}} = -\frac{2}{\mathcal{K}} e_{ijk} c_{ilk} + \frac{\lambda}{|\mathbf{X}|} * sgn(d_{jlk}) \quad (5.24)$$

Finally, we define the update of the parameters:

$$c_{ilk}^{(t+1)} = c_{ilk}^{(t)} - \eta \frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial c_{ilk}} \quad (5.25)$$

$$d_{jlk}^{(t+1)} = d_{jlk}^{(t)} - \eta \frac{\partial \mathcal{O}(\theta; X_{ij})}{\partial d_{jlk}} \quad (5.26)$$

The bias update is performed accordingly to equations (5.26).

L1 regularisation

For the L_1 case, they way the update of the parameters unfolds is different, since the sign of the regularization term in the gradient depends on the sign of the parameter. This means that the direction of the regularisation component changes as we pass from one side of the axis *parameter* $= 0$ to the other. Thus, to implement the gradient step properly (and ensure that the learnt model is sparse), we need to prevent the gradient step from crossing the axis and changing the sign of the parameter. This can be done by simply checking whether the sign of the parameter has swapped (from -1 to 1 or vice-versa) after the update, and setting the parameter to zero if it has:

$$\text{if } [sgn(a_{ik}^{(t+1)}) == -sgn(a_{ik}^{(t)})] \text{ then } a_{ik}^{(t+1)} := 0 \tag{5.27}$$

Pseudo-code for the Gradient Descent Algorithm

Algorithm 1 provides pseudo-code implementing the SGD search for the parameters: $\theta = \{A, B, C, D\}$, $A \in \mathbb{R}^{\mathcal{IK}}$, $B \in \mathbb{R}^{\mathcal{JK}}$, $C \in \mathbb{R}^{\mathcal{ILK}}$ and $D \in \mathbb{R}^{\mathcal{JLK}}$ of the linear model $Y_{ij} = A_i + B_j + C_i^T D_j$. Note that in the algorithm are reported both L1 and L2 regularization parameters but, depending on how many parameters/latent factors will be used in the actual implementation we will decide weather or not to exploit just the Ridge regularisation term. Moreover, along with the regularization, both initialization of the parameters and convergence criteria of the stochastic gradient descent will be analyzed more in depth into the experiment and result chapter 6.

Algorithm 1 Stochastic Gradient Descent for Matrix Factorization

```
Input: $\mathbf{X}^{(train)}, \lambda, isLasso, \eta^{(0)}, maxIterations$
Output: $\theta = \{A, B, C, D\}$
 1: $A^{(0)}, B^{(0)}, C^{(0)}, D^{(0)} \leftarrow Initialize$
 2: $t \leftarrow 0$
 3: while $t < maxIterations$  &&  $notConverged$ do
 4:     for all $X_{ij} \in \mathbf{X}^{(train)}$ do
 5:         for $k \in \{1, ..., \mathcal{K}\}$ do
 6:             $e_{ijk} \leftarrow (x_{ijk} - (a_{ik} + b_{jk}))$
 7:             $a_{ik}^{(t+1)} \leftarrow a_{ik}^{(t)} - \eta^{(t)} \frac{\partial \mathcal{O}(\theta^{(t)}; X_{ij})}{\partial a_{ik}}$
 8:             $b_{jk}^{(t+1)} \leftarrow b_{jk}^{(t)} - \eta^{(t)} \frac{\partial \mathcal{O}(\theta^{(t)}; X_{ij})}{\partial b_{jk}}$
 9:             $c_{ilk}^{(t+1)} \leftarrow c_{ilk}^{(t)} - \eta^{(t)} \frac{\partial \mathcal{O}(\theta^{(t)}; X_{ij})}{\partial c_{ilk}}$
10:             $d_{jlk}^{(t+1)} \leftarrow d_{jlk}^{(t)} - \eta^{(t)} \frac{\partial \mathcal{O}(\theta^{(t)}; X_{ij})}{\partial d_{jlk}}$
11:             if $isLasso$ then
12:                 if $sgn(a_{ik}^{(t+1)}) == -sgn(a_{ik}^{(t)})$ then
13:                     $a_{ik}^{(t+1)} \leftarrow 0$
14:                 end if
15:                 if $sgn(b_{jk}^{(t+1)}) == -sgn(b_{jk}^{(t)})$ then
16:                     $b_{jk}^{(t+1)} \leftarrow 0$
17:                 end if
18:                 if $sgn(c_{ilk}^{(t+1)}) == -sgn(c_{ilk}^{(t)})$ then
19:                     $c_{jlk}^{(t+1)} \leftarrow 0$
20:                 end if
21:                 if $sgn(d_{jlk}^{(t+1)}) == -sgn(d_{jlk}^{(t)})$ then
22:                     $d_{jlk}^{(t+1)} \leftarrow 0$
23:                 end if
24:             end if
25:         end for
26:     end for
27:     $t$++;
28: end while=0
```

5.4 Tensor factorization

As an alternative to the kNN and MF approaches, the state of the art model PREDICTD makes use of a PARAFAC tensor decomposition to predict the unobserved signal Y_{ij} by factorizing the training data $\mathbf{X}^{(train)}$. As already explained in Section 2.4 they do employ three factor matrices, one per dimension, and three matrices of biases with one values per component in the different dimensions.

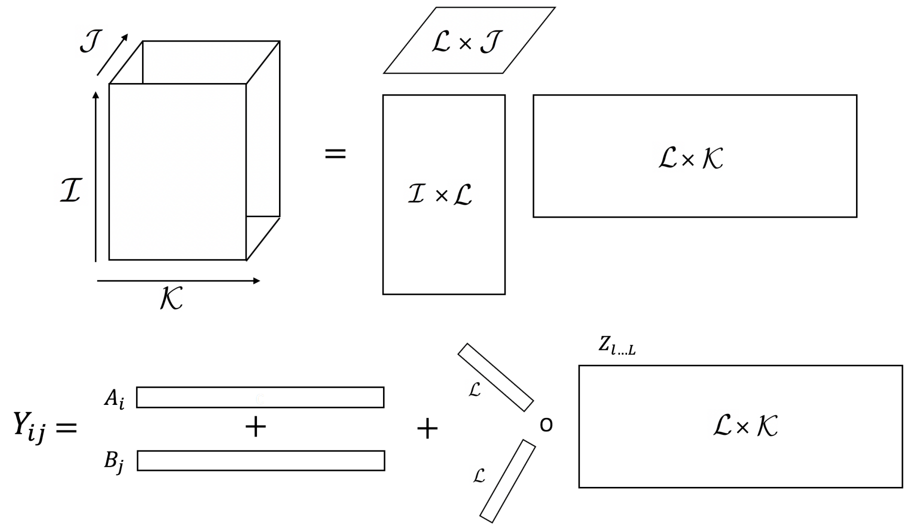

$\mathcal{J}$
$\mathcal{I}$
$\mathcal{K}$
$=$
$\mathcal{L} \times \mathcal{J}$
$\mathcal{I} \times \mathcal{L}$
$\mathcal{L} \times \mathcal{K}$
$Y_{ij} =$
A_i
B_j
$+$
$+$
$\mathcal{L}$
$\mathcal{L}$
$\circ$
$Z_{l \ldots L}$
$\mathcal{L} \times \mathcal{K}$

Figure 5.3: Tensor factorization model

For the ENCODE problem, where the genome dimension is many orders of magnitude longer than the other dimensions (indeed, the provided tensor describes 700 signals less than the one used by PREDICTD), it is easier and more meaningful to interpret the decomposition trying to identify a number of latent signals that characterize the whole training set tracks and that can be linearly combined based on the current couple cell-assay for which we want to predict the test instance. More formally, the decomposition comes as a weighted combination of latent signals $\{Z_1, ..., Z_{\mathcal{L}}\}, Z_l \in \mathbb{R}^{\mathcal{K}}$ (Figure 5.3):

$$Y_{ij} = \sum_{l=1}^{\mathcal{L}} w_l^{(i,j)} Z_l = \sum_{l=1}^{\mathcal{L}} a_{il} b_{jl} Z_l \tag{5.28}$$

Where the weight $w_l^{(i,j)} = a_{il} b_{jl}$ used to combine the signals is assumed to be estimated during the training phase along with the latent signal parameters. Thus, we see that the PARAFAC tensor decomposition is in fact predicting an unobserved signal as a linear combination of latent signals that need to be estimated. This is not that dissimilar from the kNN model, but in this case the signals being composed are latent and must be learnt from the data. Moreover, we can extend the basic PARAFAC tensor decomposition model by adding the bias components. The biases, in this case, are not just a single value for each assay-cell-base pair like in the PREDICTD model but are structured differently, equal to the ones that we used in the Matrix Factorization model. It is also worth noticing that in this case, unlike the MF, the parameters of the factor matrices are not independent from each other and it is, indeed, possible to extract

potentially useful information on the training set tracks by analyzing the latent signals. On top of that (given also the length of each latent track), can be very beneficial to induce sparsity by introducing the L_1 regularization on the latent signals Z_l, in order to make each additive latent signal as simple (and thus interpretable) as possible.
So, we can write the new biased tensor factorization model as:

$$Y_{ij} = Z_i^{(1)} + Z_j^{(2)} + \sum_{l=1}^{\mathcal{L}} a_{il} b_{jl} Z_l^{(3)} \tag{5.29}$$

where we have used Z with different superscripts to denote the three types of signals (assay bias, cell-line bias and latent signal) that are combined to produce a prediction.

5.4.1 Tensor factorization parameter estimation

Considering always the same objective function (Equations (5.3), (5.4)), the MSE for the true signal X_{ij} and the predicted signal Y_{ij} can be written as:

$$\text{MSE}(X_{ij}, Y_{ij}) = \frac{1}{\mathcal{K}} \sum_k (x_{ijk} - (z_{ik}^{(1)} + z_{jk}^{(2)} + \sum_{l=1}^{\mathcal{L}} a_{il} b_{jl} z_{lk}^{(3)}))^2 \tag{5.30}$$

Where $z_{ik}^{(1)} \in Z_i^{(1)}$, $z_{jk}^{(2)} \in Z_j^{(2)}$ adn $z_{lk}^{(3)} \in Z_l^{(3)}$. For what concerns the computation of the gradients, what differs from the matrix factorization approach are the partial derivatives with respect to the parameters a_{il} and b_{jl}. Taken as example the one related to a_{il}, the derivative for both L1 and L2 regularization can be expressed as follows:

$$\frac{\partial O(\theta; X_{ij})}{\partial a_{il}} = -\frac{2}{\mathcal{K}} b_{jl} \sum_k^{\mathcal{K}} e_{ijk} z_{lk}^{(3)} + 2\frac{\lambda}{|\mathbf{X}|} a_{il} \tag{5.31}$$

$$\frac{\partial O(\theta; X_{ij})}{\partial a_{il}} = -\frac{2}{\mathcal{K}} b_{jl} \sum_k^{\mathcal{K}} e_{ijk} z_{lk}^{(3)} + \frac{\lambda}{|\mathbf{X}|} * sgn(a_{il}) \tag{5.32}$$

The same goes for the b_{jl} parameters:

$$\frac{\partial O(\theta; X_{ij})}{\partial b_{jl}} = -\frac{2}{\mathcal{K}} a_{il} \sum_k^{\mathcal{K}} e_{ijk} z_{lk}^{(3)} + 2\frac{\lambda}{|\mathbf{X}|} b_{jl} \tag{5.33}$$

$$\frac{\partial O(\theta; X_{ij})}{\partial b_{jl}} = -\frac{2}{\mathcal{K}} a_{il} \sum_k^{\mathcal{K}} e_{ijk} z_{lk}^{(3)} + \frac{\lambda}{|\mathbf{X}|} * sgn(b_{jl}) \tag{5.34}$$

The derivatives for the other parameters, biases and latent signals are computed the same as for the matrix factorization model (Equations (5.10),

(5.21)).
Specific explanations for the initialization, training and convergence of all the models presented in this chapter will be given in the following Chapter 6.

Chapter 6

Experiments and results

6.1 Experiments set up

In this chapter I will describe the different experiments performed and the relative results obtained.
The first step is to clarify what data format we did use in order to perform the experiments and how we got there. As already specified, the dataset is a tensor with dimensions $\mathcal{I} \times \mathcal{J} \times \mathcal{K}$ where, for each couple (i,j), corresponds a genome wide signal X_{ij}. These signals were provided by ENCODE in the *bigWig*, format so we had to download them and convert them into the *bedGraph* format and then into *numpy* format (see section 3.2). Each array corresponds to a signal with 3 billion positions so, we had to downsample them at the 25 base pair resolution in order to make them fit into memory and to drastically reduce the computation time for the different experiments. Further transformations of the data have been done, but they will be analyzed in relation to the particular technique they are used for.
In this chapter I will report sequentially pairs of experiments with their relative results, partitioning them into three macro-groups, each one relative to the respective model: (i) kNN regressor (ii) additive plus matrix factorization model (iii) tensor factorization model. Reported all the relevant results and how we have obtained them, I will proceed to confront the different methods, comparing the results and trying to answer the different research questions proposed in chapter 2. At the end I will also mention the results we managed to achieve in the ENCODE imputation challenge, what predictions submitted before the deadline, and confront the local results with the challenge results.
All the different performances will be expressed through out the relative MSE measure reported in chapter 3, averaging off the relative MSE for all the target tracks in order to get a global measure, both for validation and test sets. Indeed, we thought this could have been the best metric to present the different performances, allowing us to be unbiased towards

outliers when delivering a global measure for one technique and, at the same time, to understand the performances relative to a fixed baseline. Here is again formalized the relative MSE computation:

$$\text{relMSE}(X_{ij}, Y_{ij}; Y_{ij}^{(base)}) = \frac{\text{MSE}(X_{ij}, Y_{ij})}{\text{MSE}(X_{ij}, Y_{ij}^{(base)})} \tag{6.1}$$

Other than the relative MSE, for each prediction, we reported also seven of the remaining measures used in the challenge, presented in section 3.4 (also in this case we will refer to performances relative to the baseline). The considered baseline is the performance of the Avocado model; indeed, it is the one which was declared to achieve the best performance among the ones known from the literature. Moreover, ENCODE provided the predictions on the validation tracks with the model trained on just the training set and the predictions on the test tracks with the model trained with training and validation sets, allowing us to perform local comparisons with our algorithms.

6.2 KNN regression

In this section I will refer to experiments and results related to the kNN regression model. The three main prediction methods that will be discussed are the assay prediction (4.7), the cell prediction (4.8) and the merge based on aggregation of predictions (4.23) (the merge based on the concurrent combination of assay and cell predictions (4.22) has been consistently outperformed across different evaluations so will not be reported).
These different methods will be analyzed with data at the original scale and then transformed in order to understand if there actually are meaningful differences between the approaches. The similarity values used, in order to perform the weighted sum of the signals for the assay and cell prediction, are the estimated ones between different cell-lines and different assay-lines (Figure 4.3(a), 4.3(b)). In the whole process, I will firstly show the results related to the validation set signals, combining the training signals, in order to show the main steps and decisions that led towards the best technique found; secondly, I will present and confront the results of the best methods got so far on the test set, combining both training and validation tracks.

6.2.1 Assay and Cell based prediction

The evaluations presented in Figure 6.1 and Table 6.1 represent the results obtained with the assay and cell based predictions expressed through the different relative measures. According to their equations, we have performed the weighted sum of all the signals related to the assay line j or cell line i, exploiting the estimated similarities.

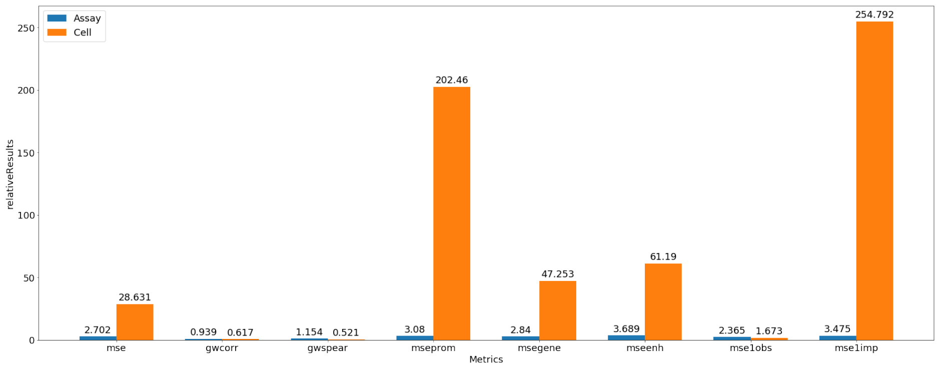


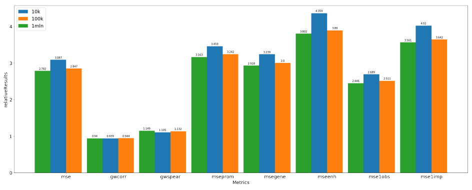
Figure 6.1: Assay and cell-based predictions

		Assay	**Cell**
Metric	MSE	2.702	28.631
	gwcorr	0.939	0.617
	gvspear	1.154	0.521
	MSEprom	3.08	202.46
	MSEgene	2.84	47.253
	MSEenh	3.689	61.19
	MSE1obs	2.365	1.673
	MSE1imp	3.475	254.792

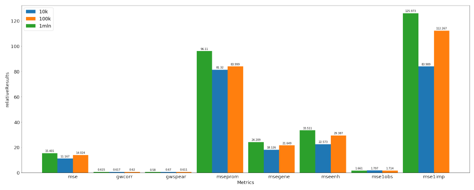
Table 6.1: Performance of Assay and Cell-based predictions

The signals are not normalized and it is immediately clear that the assay based predictions consistently outperform the cell based predictions both for the correlation measures and the MSE measures. The outcome of these evaluations confirms the assumptions made during the analysis of the distribution of the signals (section 3.2); indeed, the bigger difference in scale existing between the signals related to the same cell i and different assays j influences negatively the predictions. The similarity value is, in fact, not able to fully capture the extent of the scale difference, making the predictions biased towards the outliers with huge values. Moreover, this type of results confirms the necessity of weighting through the k most similar signals, without exploiting all the tracks available on the current cell-line i.

Assay and Cell-based prediction at different resolutions



(a) Assay (10k, 100k, 1mln)



(b) Cell (10k, 100k, 1mln)

Figure 6.2: Assay and Cell prediction at different resolution

(a)

		10k	100k	1mln
Metric	MSE	3.087	2.847	2.782
	gwcorr	0.939	0.944	0.940
	gvspear	1.105	1.132	1.149
	MSEprom	3.459	3.242	2.928
	MSEgene	3.239	3.0	3.802
	MSEenh	4.359	3.89	3.802
	MSE1obs	2.689	2.511	2.446
	MSE1imp	4.02	3.642	3.561

(b)

		10k	100k	1mln
Metric	MSE	11.167	14.024	15.401
	gwcorr	0.617	0.620	0.615
	gvspear	0.670	0.611	0.58
	MSEprom	81.320	83.999	96.110
	MSEgene	18.126	21.649	24.209
	MSEenh	22.573	29.387	33.511
	MSE1obs	1.797	1.714	1.661
	MSE1imp	83.989	112.267	125.973

Table 6.2: (a) Assay-based predictions (b)Cell -based predictions at different resolutions (10k, 100k, 1mln)

To investigate further in detail the assay and cell predictions we tried to exploit the similarities across the genome at different resolutions (4.17). Understanding weather or not we were generalizing too much adopting global similarities for signals with a considerable amount of positions was an important step to take in order to be sure about committing time into the development of the correct algorithm. So, given three different resolutions for the 25 base pair signals (10k, 100k, 1mln), we estimated the cell-cell similarity and assay-assay similarities exploiting the distinct segments coming from the partitions. The prediction has been performed by weighting the corresponding segments of the genome, at the chosen resolution, using the respective similarity matrices.
The outcome of the experiments (Figure 6.2, Table 6.2) suggests that performing locally the predictions improves significantly the cell based predictions. Indeed, specially for the 10k resolution case there are big improvements across all the different measures. Instead, from the assay prediction point of view, so far still the best one, there are no improvements. There is in fact a slightly decrease in performances with respect to the global prediction.
In order to check if it was necessary to explore the similarities at even smaller resolutions or continue committing on the development of the

genome-wise techniques, we decided to investigate weather or not merging ”local similarity” predictions could outperform merging ”global similarity” predictions.

Introducing the Merge prediction

Given the results of the assay and cell based prediction, from global to local similarities, I will present now the outcome of the evaluation performed with the merge imputed tracks.
First of all, it is important to compare, at global resolution, the results between the assay prediction and merge prediction trying to understand if the latter can deliver more accurate imputations that the former. Looking at the plot 6.3 it is clear how, except for the MSEprom and MSE1imp, the merge technique delivers better imputations than the assay based prediction. The results considered to make the comparison are relative to tracks imputed according to the linear combination of the same assay and cell predictions considered in Table 6.1 in order to ensure a meaningful comparison between the techniques. The α coefficient of equation (4.23) has been computed, for each target track Y_{ij}, according to equation (4.25) with K_i and K_j sets computed with the top-k% = 20% (equation (4.27).

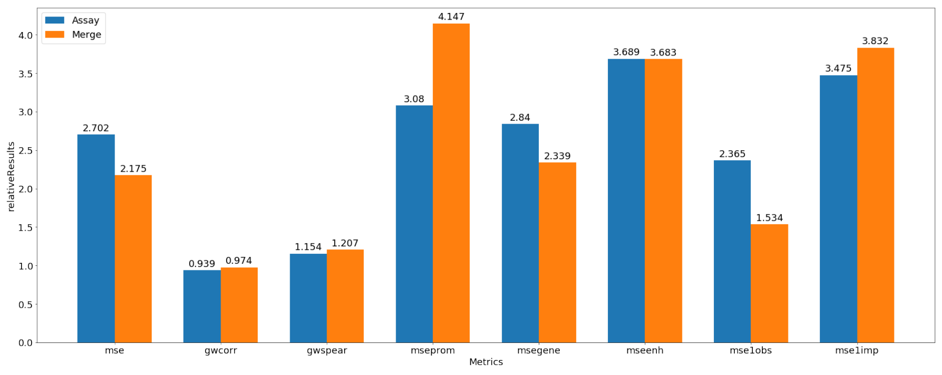


Figure 6.3: Assay and Merge predictions

From this point and up to section 6.2.3 all the results related to the merge technique will refer to the same top-k% = 20% parameter used to compute the α aggregation coefficient.
Since the above merge technique is, so far, the best one according to the relative metrics, we confronted its results with the performances related to the imputed signals produced through out the merging of assay and cell predictions at different resolutions.
Looking at the Table 6.3, it is possible to understand how, lowering the resolution from 1 million base pairs to 10 thousand base pairs, the results get worse and, also for what concerns the comparison between the global merge and 1 million resolution merge, there are not relevant differences,

so, we stopped investigating further into local similarities between signals. The lack of meaningful differences in the evaluation is probably due to the fact that the signals are mainly flat; indeed, lowering the resolution would not increase the information available but just continue computing more and more similarities between vectors with zero standard deviation.

		Merge			
Settings	Resolution	Global	1mln	100k	10k
Metric	MSE	2.175	2.20	2.253	2.436
	gwcorr	0.974	0.975	0.976	0.962
	gvspear	1.207	1.215	1.198	1.192
	MSEprom	4.147	4.045	4.155	4.696
	MSEgene	2.339	2.342	2.401	2.625
	MSEenh	3.683	3.741	3.8	4.194
	MSE1obs	1.534	1.578	1.632	1.765
	MSE1imp	3.832	3.758	3.876	4.31

Table 6.3: Merge between assay and cell predictions at different resolutions

Also in this case, the merge techniques for the 10k, 100k and 1mln resolutions have been performed with same process and same hyperparameters as the global merge one, ensuring a meaningful comparison between the results.
Given the outcome of the comparison, from now on, all the considered predictions will refer only to global weighting of the signals.

6.2.2 Transformed signals

As already explained in the relative section 3.2, three different transformation techniques have been applied to the dataset in order to solve the problems related to the differences in scale between the signals: $\sinh^{-1}$, log, and $\log\log$. In the following section, I will report the results obtained with the best technique found so far (merge prediction) according to the different transformation of the dataset. The final aim is to understand which one of the different functions can lead to better results and, consequentially, fits better the values of the signals.
The results obtained with the merge technique are reported in the Table 6.4 and it is clear how the accuracy of the prediction drastically improves. Indeed, the MSE, which is the main metric we looked at, goes from 2.175 down to 0.976. So, we can assume that all the three transformations solve the problem related to the difference of scale in the dataset. Indeed, the big improvement in performance can be also justified by the fact that the KNN techniques work much better when the data is normalized and the

values are on the same scale.
The log and log log transformations are worse than the $\sinh^{-1}$. The log log flattens all the signals for values greater than 25, solving the problem of large values, but, at the same time, canceling meaningful differences in scale between the active regions. The log is quite similar to the $\sinh^{-1}$ transformation but the difference in performance can be brought back to the fact that, even if they penalize in the same way high values and low values, they treat differently the numbers in the range from ~5 to ~25, which correspond to the meaningful values for the majority of the signals in the dataset.

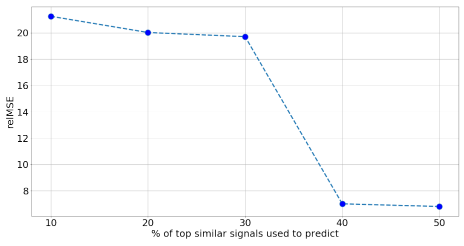
		Merge			
Settings	Transformation	-	$\sinh^{-1}$	*log*	*loglog*
Metric	MSE	2.175	0.976	1.176	1.278
	gwcorr	0.974	1.094	1.005	942
	gvspear	1.207	1.179	1.165	1.130
	MSEprom	4.147	0.957	1.173	1.291
	MSEgene	2.339	0.964	1.174	1.28
	MSEenh	3.683	0.980	1.157	1.27
	MSE1obs	1.534	0.991	1.272	1.389
	MSE1imp	3.832	0.985	1.241	1.375

Table 6.4: Merge between predictions made with different transformation of the signals

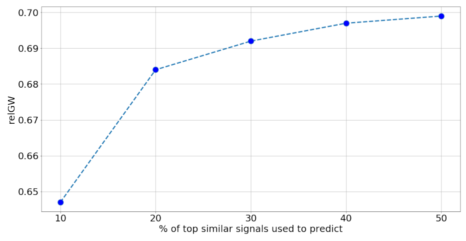
Given the fact that the $\sinh^{-1}$ allowed us to obtain, for each one of the different metrics, results on the validation set better than the Avocado baseline, we can assume with relative confidence that this is the transformation that better fits the data. Hence, every evaluation from now on will make use of a $\sinh^{-1}$ transformed dataset, both for KNN and factorization models. This kind of function has already been used by Avocado and PREDICT in order to transform their data; however, the signals they worked on were generated by exploiting different assays and with a slightly different process (the depth of the different reads was different in the sequencing techniques, section 2.3), so the comparison between the different functions, applied to our data, could have still revealed meaningful differences.
Since all the attempts made with the rescaling techniques, which were aimed at solving the same problem, as presented in subsection 4.2.3, were outperformed by the implementation of the $\sinh^{-1}$ transformation, they will not be mentioned in this result section.

6.2.3 Sweep over the parameters

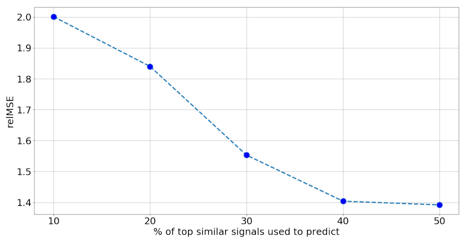
Established the correct transformation for the data, the last step is to research the correct number of k nearest neighbour that needs to be used to impute the tracks for assay and cell-based predictions. Tuning the number k can, in fact, lead to better merged prediction results. Indeed, especially for same cell-line signals, even if there are not anymore big differences in scale, there might still be relevant dissimilarities between different assays like *DNase-seq* and *Chip-seq*.



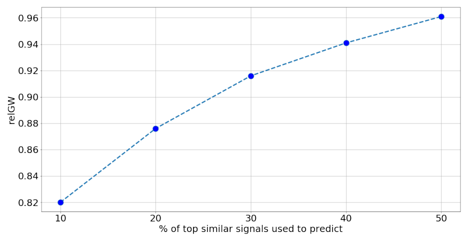
(a) Cell: relMSE sweep 10 to 50%



(b) Cell: relGWcorr sweep 10 to 50%



(c) Assay: relMSE sweep 10 to 50%



(d) Assay: relGWcorr sweep 10 to 50%

Figure 6.4: Assay and Cell top k nearest neighbours sweep

Thus, starting from the cell prediction, we did sweep for different number of k, expressing the percentage of signals involved to perform the imputation for every validation track, going from 10% to 50% top similar signals along the current $i \in \mathcal{I}$. In order to check the outcome of the sweep, we did plot the results for the two main metrics of the two macro-groups: relative MSE and relative GWcorrelation (Figures 6.4(a), 6.4(b), respectively). The same process has been done for the assay predictions, also sweeping from 10% to 50% of the top similar signals along the line $j \in \mathcal{J}$.
For what concerns the cell prediction, $k = 50\%$ leads to the best results found so far, which are also better than the $k = 100\%$ cell based prediction with $\sinh^{-1}$ transformation on the data. This outcome confirms the fact that using the top-k most similar signals can allow to achieve better results, avoiding to introduce noise in the predictions due to natural differences between the different types of assays relative to the same cell. Analyzing,

instead, the sweep for the assay prediction, even if the k= 50% give the best results, it still does not outperform the k=100% assay prediction with $\sinh^{-1}$ transformation, showcasing how it is, in any case, more valuable to use all the information available along the assay-line dimensions.
Comparing the previous merge prediction with the new one, which combines the best top-k assay and cell predictions and leaving unchanged the parameters for the aggregation, we indeed notice how the performances on the validation increase on all the different relative evaluation metrics (Table 6.5).

		Merge	
Settings	Assay-Cell top-k	100%-100%	100%-50%
Metric	MSE	0.976	0.950
	gwcorr	1.094	1.096
	gvspear	1.179	1.175
	MSEprom	0.957	0.948
	MSEgene	0.964	0.938
	MSEenh	0.98	0.962
	MSE1obs	0.991	0.945
	MSE1imp	0.985	0.968

Table 6.5: Merge comparison

Other than the sweep on the assay and cell predictions, we tried also to perform a sweep over the parameters used to compute the merge prediction. Up to now, in order to determine the α parameter for the aggregation, we defined the K_i and K_j set of signals just considering the 20% more similar ones across the corresponding cell line i and assay-line j. Thus, we performed a sweep on the top-k% parameter used to build the K_i and K_j sets, going from the 10% top similar signals to the top 50% similar signals, in order to understand weather or not, changing the way the predictions are dynamically aggregated, positively affects the accuracy of the predictions. After the sweep, the result confirmed that the top 20% for K_i and K_j set of signals is the optimal parameter. So, given the outcome of the sweep, we can consider the result in table 6.5 as the best one related to the KNN regressor. Moreover, the final result on the validation does indicate that, looking at relMSE, we are 5% better than Avocado, which is a good result at support to our approach.
Here we have the Table 6.6 that resumes all the best results relative to the different techniques and parameters presented so far, including also the best method.

		Merge			
Settings	Assay-Cell top-k	100%-100%	100%-100%	100%-100%	100%-50%
	Resolution	Global	1mln	Global	Global
	Transformation	-	-	sinh -1	$\sinh^{-1}$
Metric	MSE	2.175	2.200	0.976	0.950
	gwcorr	0.974	0.975	1.094	1.096
	gvspear	1.207	1.215	1.197	1.175
	MSEprom	4.147	4.045	0.957	0.948
	MSEgene	2.339	2.342	0.964	0.938
	MSEenh	3.683	3.741	0.980	0.962
	MSE1obs	1.534	1.578	0.991	0.945
	MSE1imp	3.832	3.758	0.985	0.968

Table 6.6: Comprehensive table for the kNN merge results

6.3 Factorization models

The experiments for the factorization models have all been carried out as follows. First I train on the training set and predict on the validation set to tune the hyperparameters. Second, fixed the hyperparameters, I retrained on training set plus validation set signals in order to predict for the test set instances. It is important to mention that, in order to reduce the required computation time, the experiments related to tuning hyperparameters have not been performed on the entire tracks but on a randomly selected subset of one million of base pairs selected from all the chromosomes. This type of approach is similar to the one adopted by Avocado and PREDICTD (section 2.4) which, in fact, only used the ENCODE Pilot regions [21] genomic locations in order to represent the signals, train and make predictions with their model. Those correspond to 1% of the genome length, that is about 1 million base pairs at 25 bp resolution. Thus, since we did sample about the same number of base pairs and half of the Pilot regions are also randomly selected, we believe that the results obtained through the sampled signals are reliable to obtain an unbiased estimate of performance for selecting the hyperparameter values. Moreover, executing several times the training of the model to sweep over the hyperparameters, exploiting the signals at 25 base pairs resolution, is unfeasible from the computational time point of view (considering the current implementation and approach). Training the matrix factorization model with the complete signals takes, in fact, around 30h to converge with a relatively high learning rate parameter.
The technique used to train the models, as already mentioned in chapter

5, is the stochastic gradient descent, which checks at each epoch the MSE on the training set and the relative MSE on the validation set to make sure the model is learning information and converging to an optimum. For what concerns the iterative method, it is also important to mention that for all the training phases, in order to converge to the best result without the risk of overfitting, we used the early stopping technique [31]. Hence, when training on the training set to tune the hyperparameters, we stopped training when the relative MSE on the validation set started increasing. When training with both training and validation set to predict the test set signals, we stopped when the improvement of the relative MSE on the validation set, between two consecutive epochs, turned negative or fell below than 0.001.

Before starting the training phase, we initialized all the parameters for additive, matrix and tensor factorization, accordingly to the same method used by the state-of-the-art PREDICTD model, with values extracted from an uniform distribution between -0.33 and $+0.33$.

In the following subsection we will analyze the results obtained, through the hyperparameters tuning on the validation set, for the additive and matrix factorization models.

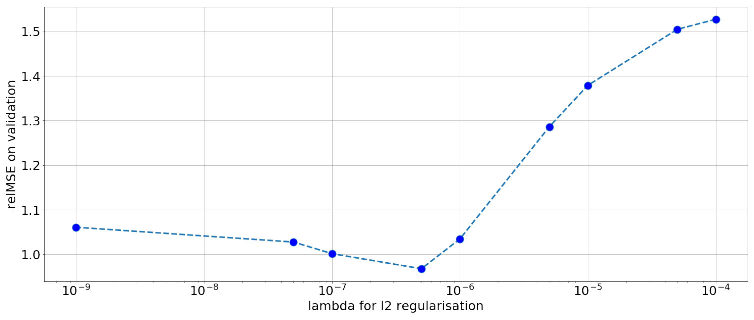
6.3.1 Hyperparameter tuning

Here, we will analyze the hyperparameters tuning for the additive and matrix factorization models. In order to perform the optimization task, we employed the grid search technique [32].

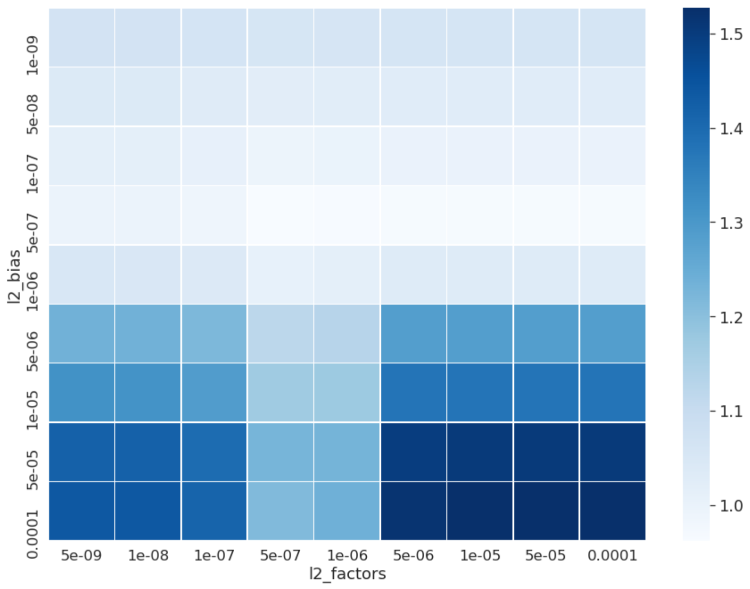
Prior to the sweep, a learning rate η equal to 10^6 was chosen for all the factorization techniques and its decay epoch by epoch can be expressed like follows: $\eta^{t+1} = \eta^t * \eta_{decay}$ with $\eta_{decay} = 1 - 1e^{-6}$.

For the additive model (equation (5.5)) we used Ridge regression regularization terms with one coefficient for both assay and cell biases parameters, λ_{bias}. We searched the hyperparameter space for $\lambda_{bias} = 1 * 10^{-i}$ or $5 * 10^{-i}$ with $i \in [4..9]$, and the best relMSE value found is 0.95 for $\lambda_{bias} = 5e^{-7}$ (Figure 6.5(a)).

For the matrix factorization model (equation (5.16))we optimized as follows: Firstly we set the number of latent factors $\mathcal{L}$ equal to 2 (we saw that increasing then number of latent factors was decreasing the performance of the model). Secondly, we decided to employ, always with Ridge regularization, two different regularization coefficients: one for the biases and one for the latent factor parameters, respectively λ_{bias} and λ_{factor}.



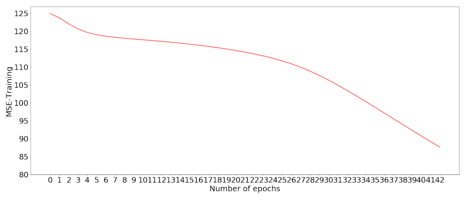
(a) Additive model λ_{bias} sweep. Best $\lambda_{bias} = 5e^{-7}$



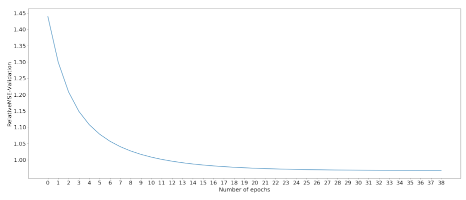
(b) Matrix factorization model $(\lambda_{bias}, \lambda_{factor})$ sweep. Best $(\lambda_{bias}, \lambda_{factor}) = (5e^{-7}, 5e^{-7})$

Figure 6.5: Sweep for the λ^{Ridge} regularization hyperparameters

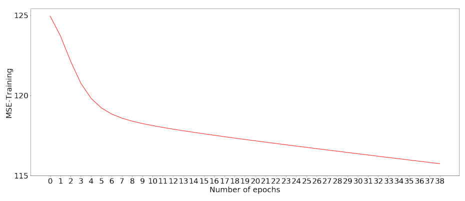
The hyperparameter space for both coefficients was equal to the one of the additive model; indeed, $\lambda_{bias}, \lambda_{factor} = 1*10^{-i}$ or $5*10^{-i}$ with $i \in [4..9]$ (Figure 6.5(b)). The best relative MSE found is 0.96 for $\lambda_{bias} = \lambda_{factor} = 5e^{-7}$, which is also in this case around 5% better than the Avocado model. In the following graphs (Figure 6.6) it is possible to see the trending of both relative MSE on the validation set and MSE on the training set for the additive model and matrix factorization best runs, respectively with 39 and 43 epochs necessary to converge (until relative MSE started increasing).



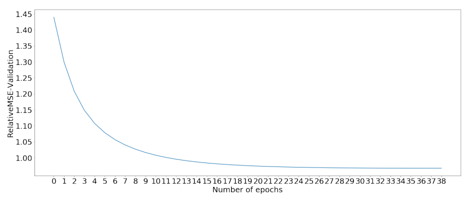
(a) Additive: MSE (Train)



(b) Additive: relMSE (Validation)



(c) MF: MSE (Train)



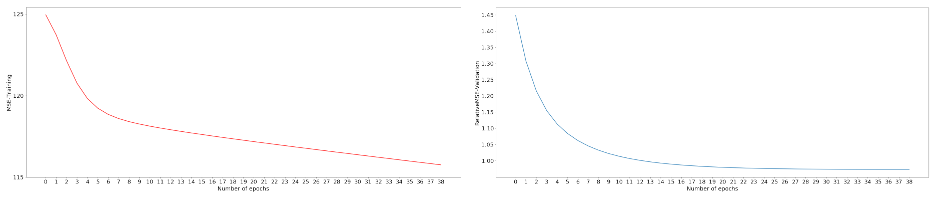
(d) MF: relMSE (Validation)

Figure 6.6: Trend of MSE on training and relative MSE on validation set for Additive and Matrix factorization best runs

6.3.2 Tensor factorization

In order to set hyperparameters for the tensor factorization model we also used the one million length samples of the full genome tracks at 25 base pair resolution, still stopping the training once the relative MSE on the validation started increasing. Given the parameters used to impute the signals (equation (5.28)) we decided to use two coefficients λ_{bias}^{Ridge} and $\lambda_{factors}^{Ridge}$ for the Ridge regularization, adding a third one λ_{latent}^{Lasso}, coefficient for the Lasso regularisation term, applied exclusively for the latent signals parameter update. Indeed, as already explained in section 2.6, given the high number of base pairs latent factor parameters, the idea was to add a regularisation term, the elastic net in this case, to increase the sparsity in the latent signals, also making an eventual interpretation of the results easier and more meaningful.
For the training phase, due to the nature of the gradients computed with respect to the assay and cell factors (equation (5.32)), we had to adapt the learning rate η dependently on the parameters we were going to update. Indeed, we set $\eta = 10^6$ for the biases and latent signals updates and $\eta = 1$ for the assay and cell factors update; the learning rate decay has remained the same for both: $\eta^{t+1} = \eta^t * \eta_{decay}$ with $\eta_{decay} = 1 - 1e^{-6}$.
The tuning of the remaining hyperparameters, λ_{bias}^{Ridge}, $\lambda_{factors}^{Ridge}$, $\lambda_{factors}^{Lasso}$ and number of latent factors $\mathcal{L}$ has been performed manually, obtaining the best run with the followings: $\mathcal{L} = 5$, $\lambda_{bias}^{Ridge} = \lambda_{factors}^{Ridge} = 5e^{-7}$ and $\lambda_{factors}^{Lasso} = 1e^{-6}$. Along with those hyperparameters the training task converged at relative MSE = 0.97 on the validation, still performing better that the Avocado baseline.

Looking at the graphs 6.7 it is possible to see the trend with which MSE for the training set and the relative MSE for validation were decreasing before converging.



(a) TF: MSE (Train) (b) TF: relMSE (Validation)

Figure 6.7: Trend of relative MSE on validation and MSE on training set for tensor factorization best run

6.4 Test set results

In this section we will analyze and compare the result obtained on the test set instances by predicting through out the different models explained so far. Also, to evaluate the performance on the test set, we will still use all the metrics introduced by ENCODE.

6.4.1 KNN results

The predictions with the KNN model have been executed with the same modality as for the validation set, averaging across tracks coming from both training and validation set. We will analyze the results obtained from the techniques used to predict the tracks submitted for the *Imputation Challenge*. In particular, three set of predictions, which are generated starting from $\sinh^{-1}$ transformed data (the best transformation), have been tested on the blind instances. The first one, which is the best model obtained through sweeping on the validation set, is the merge between the top-100% assay based prediction and the top-50% cell based predictions; instead, the other two have been chosen trying to reason about the nature and distribution of the test set. Indeed, the sweep over the parameters could have led us to overfit the predictions on the validation set, so we decided as well to submit the merge prediction that aggregates top-100% assay and cell based prediction, also considering that it still performs better than Avocado on the validation. Moreover, given the fact that there is very little information exploitable from the cell dimension when predicting for the test tracks, we decided to submit as well just the top-100% assay prediction. The results are in table 6.7 and will be discussed after reporting the results for the factorization models.

		Merge		Assay
Settings	Assay-Cell top-k	100%-100%	100%-50%	100%-None
Metric	MSE	0.914	0.913	0.887
	gwcorr	1.295	1.292	1.320
	gvspear	1.289	1.284	1.304
	MSEprom	0.903	0.902	0.857
	MSEgene	0.899	0.898	0.867
	MSEenh	0.882	0.882	0.876
	MSE1obs	0.984	0.982	0.958
	MSE1imp	1.518	1.513	1.484

Table 6.7: KNN test set results

6.4.2 Factorization models results

As already explained in section 6.3, the results on the the test set for the factorization models have been produced by training the models on both the training and validation set composed by genome-wide signals, exploiting the hyperparameters found (Table 6.8) on the validation set by training with the one million length subset of the tracks. Also, the technique used to converge is still the early stopping and it is again performed looking at the relative MSE on the validation set. The algorithms converge once the improvement between two consecutive epochs is negative or less than 0.001.

		Add.	**MF**	**TF**
HyperParam	η	10^6	10^6	$10^6/1$
	η_{decay}	$1-1e^{-6}$	$1-1e^{-6}$	$1-1e^{-6}$
	$\mathcal{L}$	$-$	2	5
	λ_{bias}^{Ridge}	$5e^{-7}$	$5e^{-7}$	$5e^{-7}$
	$\lambda_{factors}^{Ridge}$	$-$	$5e^{-7}$	$5e^{-7}$
	$\lambda_{factors}^{Lasso}$	$-$	$-$	$1e^{-5}$

Table 6.8: Hyperparameters recap

While the time necessary to train the models on the one million tracks was negligible, in this case on the full dataset, additive, matrix factorization and tensor factorization models training time was equal respectively to: 15h, 31h and 116h. The results obtained on the test set are summarized in Table 6.9 and will be commented and compared with the KNN ones in the following subsection.

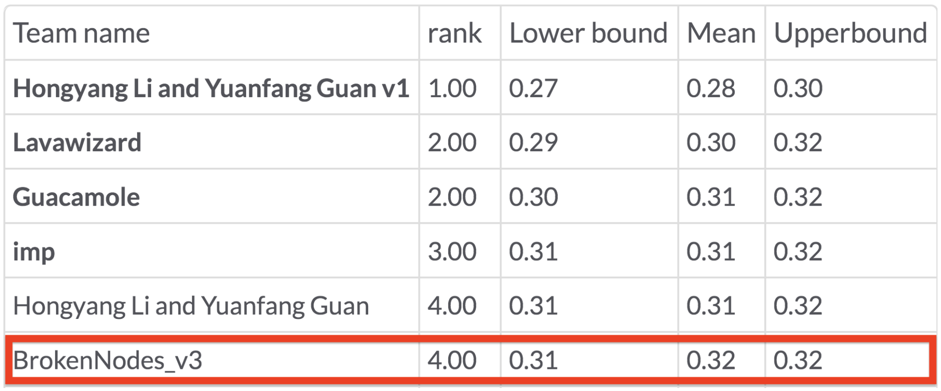
		Add.	**MF**	**TF**
Metrics	MSE	1.063	1.044	1.009
	gwcorr	0.896	0.979	1.011
	gvspear	0.757	0.730	0.727
	MSEprom	1.139	1.034	1.030
	MSEgene	1.095	1.017	1.013
	MSEenh	1.219	1.015	1.017
	MSE1obs	1.046	1.069	1.067
	MSE1imp	1.753	1.407	1.594

Table 6.9: Factorization test set results

6.4.3 Comparing KNN and Factorization models

The intuition regarding the possibility of the KNN model outperforming the Avocado baseline (section 2.6) has been proven correct. The high number of missing values plus the little information on the cell dimension for the test set tracks defines, in fact, very difficult conditions to successfully train a tensor factorization model. An instance based method has showcased to be more effective at exploiting the little information available in the data in order to make predictions on the blind set of signals. The best prediction came from the assay based model, which manged to achieve very good evaluations for both the MSE and correlation group of measures, performing 10% better than Avocado in terms of MSE (0.887) and 30% better in terms of GWcorr (1.320). The merge techniques still outperformed the baseline but got worse result than the assay based technique. Thus, the attempt to integrate the information available for the test set cell-line tracks ended up adding noise to the predictions instead of actually improving the quality.
While the KNN got good results for the specific task, the factorization models fell short, without succeeding in outperforming the Avocado predictions, even if the performances achieved by the tensor factorization are, indeed, very close (1.009 relative MSE). Anyway, given the fact that an instance based model managed to outperform a deep tensor factorization model, in order to really obtain reliable comparisons between the baseline and our implementations and then, state that our solutions for the problem do not actually represent a viable way to improve the accuracy of the imputations, it could be beneficial to try and learn the model starting from a different dataset, more similar to the one used to train and evaluate Avocado.

6.5 Imputation challenge



Team name	rank	Lower bound	Mean	Upperbound
Hongyang Li and Yuanfang Guan v1	1.00	0.27	0.28	0.30
Lavawizard	2.00	0.29	0.30	0.32
Guacamole	2.00	0.30	0.31	0.32
imp	3.00	0.31	0.31	0.32
Hongyang Li and Yuanfang Guan	4.00	0.31	0.31	0.32
BrokenNodes_v3	4.00	0.31	0.32	0.32

Figure 6.8: Challenge first four rankings. Taken from [15]

In this last section, I will report the results we managed to achieve joining and submitting for the *ENCODE Imputation Challenge*.
By the time of the challenge deadline, the only developed model was the KNN regressor, so, we submitted three different sets of predictions coming respectively from the three methods mentioned in section 6.4.1. The test set was not available before the publication of the results since the signals still needed to be generated by the challenge organizers at the time of the deadline.
According to their evaluation measure, we managed to achieve rank 4 (Figure 6.8) with the submission generated from the merge technique that aggregates the top-100% assay prediction and the top-50% cell prediction. The latter, on their ladder, also managed to outperform the assay based prediction which, with our local relative MSE evaluation measure, results instead as the top performing technique on the blind dataset. It is important to note that the prediction made through out the rank 4 technique is not quite the same as the one presented in section 6.4.1. In order to generate the prediction, we not only exploited the training and validation set but also the predictions of the Avocado model on the test set (i.e., the Avocado imputed runs which had been made available by ENCODE in order to use them). Nevertheless, according to our scoring, the improvement in performance is quite minimal and the assay prediction still remains the best method. The comparison is reported in the following Table 6.10, where the line *Signals* defines the pool of tracks used in order to make the prediction (Training T, Validation V, blind tracks imputed with a given method x B_x).

		Merge		Assay
Settings	Signals	$T, V, B_{avocado}$	T, V	T, V
	Assay-Cell top-k	100%-50%	100%-50%	100%-None
Metric	MSE	0.898	0.913	0.887
	gwcorr	1.357	1.292	1.320
	gvspear	1.319	1.284	1.304
	MSEprom	0.877	0.902	0.857
	MSEgene	0.879	0.898	0.867
	MSEenh	0.874	0.882	0.876
	MSE1obs	0.976	0.982	0.958
	MSE1imp	1.557	1.513	1.484

Table 6.10: KNN test set results comparison

Thus, the discrepancy between the two different evaluations is probably due to the different process with which ENCODE decided to score the different imputations. While our process is more straightforward, directly comparing the prediction with the baseline, the way in which they compute the performances of the different techniques submitted by the participating teams is composed of five distinct steps [15], which are as follows:

1. For each predicted signal on the blind test set all the introduced measures are calculated for 10 different bootstraps. Each bootstrap samples with replacement the chromosomes in order to obtain a signal that represents about 90% of the genome.

2. The performances of each bootstrap are evaluated with all the metrics for each couple cell-assay, for each team. Doing so, every team will have 10 different evaluations for each one of the metrics for each instance of the test set. Each one of this single evaluations is then transformed into a single score value combining the results obtained for all the metrics with the sum over all nine normalized ranking measures of ln(r/(N+1)), where r is equal to the rank obtained by the team with respect to the others for the relative metric, in one of the bootstraps on a given assay-cell prediction. Now, for each assay-cell type, all the teams will have 10 different single scores related to the different bootstraps.

3. The 10 scores, for each cell-assay, are then combined as follows to define a unique evaluation w.r.t. to the other teams: a rank is assigned based on the score of the second best bootstrap (e.g. on a given test sample the team will be first if the score associated to the

second best bootstrap is the lowest, there can be ties). This is done in order to define a ranking for each cell-assay.

4. In order to compute a unified score across all the cell-assay couple, the ranks, coming from step 2 (so 10 ranks for each cell-assay), are averaged for each bootstrap as follows: $r_i/(N+1)$ with r_i equal to the current bootstrap rank on the i-th cell-assay and N equal to the number of participants. Now, each team will have 10 different global scores that will result in 10 different ranks.

5. As in step 3 the global rank of a team will be given by the second best of the 10 ranks coming from step 4.

This procedure allows, after step 4, the possibility that a systems ranks highly because it scores well on second bootstraps despite having poor performances on all other. Indeed, considering a similar scenario, this could easily explain the discrepancy between our evaluation and their evaluation on the test set.

Chapter 7

Future work and conclusions

In this work we aimed at developing a new algorithm able to perform imputation of signals representing the activity of epigenetic markers (assays) for different cell-lines at different regions across the genome. While we wanted to introduce something novel from the modelling point of view, in order to ensure reliable performances, we also aimed at outperforming the previous state of art baseline Avocado. More specifically, the goal of the imputation task, according to the research area, consists in being able to reproduce the results of specific assays performed using sequencing epigenetics techniques on a given set of cells or tissues. To do so, we joined the *Imputation Challenge* announced by the ENCODE Consortium, getting access to a three dimensional tensor already partitioned into training, validation and blind test sets that we used in this work to train models and evaluate predictions. First, an accurate analysis of the data was performed (chapter 3) in order to understand the distribution of the values across the signals, the major differences between signals due to their natural components and if it was necessary to transform the data in order to get feasible conditions for the imputation task. Starting from the challenge, we first implemented a kNN regression method (chapter 4), believing that the relatively simple algorithms could fit the data better than factorization techniques. The implemented approaches relied on exploiting the information available on the same target signal cell-line and assay line, performing a weighted average of the selected tracks, using as similarity between signals their Pearson correlation.
After the end of the challenge, we continued to investigate other methods to decompose the dataset based on both matrix factorization and tensor factorization in order to impute with greater accuracy the blind instances proposed by ENCODE(chapter 5). The aim in developing this approaches was to try to understand weather or not, on this particular dataset, the KNN instance based method could be more effective than factorization models and the state-of-the-art Avocado method (which itself also exploits tensor factorization). The results obtained (chapter 6), showcased how the

assumption made regarding the KNN technique was correct. Indeed, it did outperform the factorization models and Avocado both on our evaluation measures, and during the challenge, managing to achieve rank 4 on the ladder.
This result is relevant because it offers a faster and more accurate technique than the factorization methods when trying to impute over dataset structured like the one provided (high sparsity and low information on the test tracks), which is, indeed, a possible scenario considering the fact that there are some cells for which we have not a consistent amount of experiments, not all the cells can be retrieved in order to perform sequencing and that, in any case, performing those techniques in the laboratory can be very expensive.
On the other hand, our factorization models, even if they achieved very close performances, did not manage to outperform the Avocado baseline. Despite this fact, since the dataset we worked on was not optimal for training tensor and matrix factorization, in order to reliably establish that the solutions we found for the factorization techniques are not optimal, would be beneficial to train and test our models with a dataset similar to the one used by Avocado. Another approach that can be related to the improvement of the factorization methods accuracy can be to try to optimize the training phase by exploiting techniques like ADAM, which could allow us to adapt dynamically the updates depending on the considered parameter, possibly improving the performances obtained.

Bibliography

[1] National Human Genome Research Institute. *What is the Human Genome Project?* URL: `https://www.genome.gov/human-genome-project/What`.

[2] Encyclopedia of DNA elements Consortium. *GenoMetric Query Language System*. URL: `https://www.encodeproject.org/help/project-overview/`.

[3] Jacob Schreiber et al. "Multi-scale deep tensor factorization learns a latent representation of the human epigenome". In: *bioRxiv* (2018). DOI: `10.1101/364976`. eprint: `https://www.biorxiv.org/content/early/2018/07/08/364976.full.pdf`. URL: `https://www.biorxiv.org/content/early/2018/07/08/364976`.

[4] Jason Ernst and Manolis Kellis. "Large-scale imputation of epigenomic datasets for systematic annotation of diverse human tissues". In: *Nature Biotechnology* 33 (Feb. 2015), 364 EP -. URL: `https://doi.org/10.1038/nbt.3157`.

[5] "PREDICTD PaRallel Epigenomics Data Imputation with Cloud-based Tensor Decomposition". In: *Nature Communications* 9.1 (2018), p. 1402. DOI: `10.1038/s41467-018-03635-9`. URL: `https://doi.org/10.1038/s41467-018-03635-9`.

[6] National Human Genome Research Institute. *Deoxyribonucleic Acid (DNA) Fact Sheet*. URL: `https://www.genome.gov/about-genomics/fact-sheets/Deoxyribonucleic-Acid-Fact-Sheet`.

[7] National Human Genome Research Institute. *NHGRI*. URL: `https://www.genome.gov/`.

[8] U.S. National library of medicine. *What is DNA?* URL: `https://ghr.nlm.nih.gov/primer/basics/dna`.

[9] U.S. National library of medicine. *What is a gene?* URL: `https://ghr.nlm.nih.gov/primer/basics/gene`.

[10] U.S. National library of medicine. *How do genes direct the production of proteins?* URL: `https://ghr.nlm.nih.gov/primer/howgeneswork/makingprotein`.

[11] "Defining functional DNA elements in the human genome". In: *Proceedings of the National Academy of Sciences of the United States of America* 111.17 (Apr. 2014), pp. 6131–6138. DOI: 10.1073/pnas.1318948111. URL: https://www.ncbi.nlm.nih.gov/pubmed/24753594.

[12] U.S. National library of medicine. *What is noncoding DNA?* URL: https://ghr.nlm.nih.gov/primer/basics/noncodingdna.

[13] U.S. National library of medicine. *What is Epigenomic?* URL: https://ghr.nlm.nih.gov/primer/howgeneswork/epigenome.

[14] National Human Genome Research Institute. *A brief guide to genomics.* URL: genome.gov/about-genomics/fact-sheets/A-Brief-Guide-to-Genomics.

[15] ENCODE Consortium. *ENCODE Imputation Challenge.* URL: https://www.synapse.org/#!Synapse:syn17083203/wiki/587192.

[16] France genomique. *Mapping of DNA-protein interaction sites: CHIP seq.* URL: https://www.france-genomique.org/technological-expertises/regulome/chip-seq-2/?lang=en.

[17] Roadmap Epigenomics. *Data processing.* URL: https://egg2.wustl.edu/roadmap/web_portal/processed_data.html.

[18] Heng Li and Richard Durbin. "Fast and accurate short read alignment with Burrows-Wheeler transform". In: *Bioinformatics (Oxford, England)* 25.14 (July 2009), pp. 1754–1760. DOI: 10.1093/bioinformatics/btp324. URL: https://www.ncbi.nlm.nih.gov/pubmed/19451168.

[19] taoliu. *MACS.* URL: https://github.com/taoliu/MACS.

[20] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014).

[21] ENCODE Project Consortium et al. "Identification and analysis of functional elements in 1% of the human genome by the ENCODE pilot project". In: *Nature* 447.7146 (June 2007), pp. 799–816. DOI: 10.1038/nature05874. URL: https://www.ncbi.nlm.nih.gov/pubmed/17571346.

[22] Ian H. Witten and Eibe Frank. *Data mining : practical machine learning tools and techniques.* Elsevier, 2005, pp. 128–136.

[23] Aggarwal and Charu C. *Recommender systems: the textbook.* Springer International Publishing, 2005, pp. 90–99. DOI: 10.1007/978-3-319-29659-3.

[24] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems”. In: *Computer* 42.8 (Aug. 2009), pp. 30–37. ISSN: 0018-9162. DOI: 10.1109/MC.2009.263. URL: https://doi.org/10.1109/MC.2009.263.

[25] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: (Sept. 2016).

[26] Tamara G. Kolda and Brett W. Bader. “Tensor Decompositions and Applications”. In: *SIAM Review* 51.3 (2009), pp. 455–500. DOI: 10.1137/07070111X. eprint: https://doi.org/10.1137/07070111X. URL: https://doi.org/10.1137/07070111X.

[27] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. “Introduction to Tensor Decompositions and their Applications in Machine Learning”. In: *ArXiv* abs/1711.10781 (2017).

[28] Tanin Sirimongkolkasem and Reza Drikvandi. “On Regularisation Methods for Analysis of High Dimensional Data”. In: *Annals of Data Science* 6.4 (Dec. 2019), pp. 737–763. ISSN: 2198-5812. DOI: 10.1007/s40745-019-00209-4. URL: https://doi.org/10.1007/s40745-019-00209-4.

[29] “The ENCODE (ENCyclopedia Of DNA Elements) Project”. In: *Science* 306.5696 (2004), pp. 636–640. ISSN: 0036-8075. DOI: 10.1126/science.1105136. eprint: https://science.sciencemag.org/content/306/5696/636.full.pdf. URL: https://science.sciencemag.org/content/306/5696/636.

[30] Univerisy of California Santa Cruz (USCS) Genomic Institute. *USCS Genome Browser*. URL: http://genome.ucsc.edu/index.html/.

[31] Lutz Prechelt. “Early Stopping - But When?” In: *Neural Networks: Tricks of the Trade*. Ed. by Genevieve B. Orr and Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 55–69. ISBN: 978-3-540-49430-0. DOI: 10.1007/3-540-49430-8_3. URL: https://doi.org/10.1007/3-540-49430-8_3.

[32] Matthias Feurer and Frank Hutter. “Hyperparameter Optimization”. In: *Automated Machine Learning: Methods, Systems, Challenges*. Ed. by Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. Cham: Springer International Publishing, 2019, pp. 3–33. ISBN: 978-3-030-05318-5. DOI: 10.1007/978-3-030-05318-5_1. URL: https://doi.org/10.1007/978-3-030-05318-5_1.