Node Duplication in Disease Maps using Graph Neural Networks

Benjamin Moser

October 20, 2021

Abstract

Contents

List of Figures

List of Tables

1 Introduction

Overview and problem statement. Probably move a few sentences to here from 2.3.

2 Background

2.1 Biological Networks

Our understanding of the molecular mechanisms that are involved in biological systems is improving drastically. However, this knowledge is growing incrementally and is often scattered across individual scientific publications. The behaviour of a biological system is often defined by complex interactions, potentially across different levels of abstraction. We refer to biological entities as *species*. Possible species types are, for instance, proteins, genes, but possibly also abstract phenotype descriptions or drugs. Possible relations include chemical reactions such as state transition (e.g. phosphorylation), physical interaction between two proteins or the effect a drug has on the function of some protein. The set of species and their interactions (relationships) naturally form a network. To properly consider complex signalling effects such as activation/inhibition, crosstalk or feedback, it is of the essence to make this network structure accessible to both human cognition and computational analysis [1].

Choices of what species and relationships to consider yield different flavours of biological networks and analyses. Since the effective biological function of proteins is rarely defined solely by their identity but rather by their roles as enzymes, signalling molecules or structural components, it is worthwhile to study *Protein-Protein Interaction Networks* (PPI networks), for instance to infer the biological function of an unknown protein or gene, or groups of functionally similar proteins. Further, an organism's metabolism may be described as a set of chemical compounds (metabolites) and the set of chemical reactions or interactions between them, yielding a *metabolic model*. Computational methods on metabolic networks can be used to predict the growth of an organism under specific conditions, identify key intervention targets or decompose the network into relatively independent subsystems. However, species and relationships need not necessarily have a direct physical counterpart. Several works investigate the interactions between diseases, drugs, proteins and their annotated biological functions [2] [3]. Moreover, biological networks may serve as a scaffold to integrate data on, e.g., gene expression or reaction rates. This can be used to classify cancer or TODO.

2.2 Disease Maps

Biochemical pathways can be described by *process description diagrams* in which species (commonly metabolites, protein complexes and genes) are linked by chemical processes. An example is given in Figure **??**. Many diseases, however, affect not only a single mechanism. Rather, a systemic understanding of the involved subsystem and their relationships is required [4][5]. Moreover, knowledge on mechanisms contributing to a disease is obtained incrementally and scattered across individual publications or database entries. Particularly for visual, interactive exploration, experts have assembled *disease maps*, comprehensive diagrams combining all known mechanisms relevant for a given disease. Traditionally, such diagrams have been drawn as pixel- or vector-based graphics. Creating and updating such graphics requries a high amount of effort and renders the contained information practically inaccessible to computational methods. Formalised, digital representations that are both human- and computer-readable provide the following immediate advantages.

- ▶ The creation process, involving the extraction of knowledge from scientific publications or databases and finding an adequate layout, may be aided by computational tools from the areas of Data Mining and Graph Drawing.
- ▶ Entities in the diagram may be annotated with additional information such as links to research publication or database entries.
- ▶ A formalised representation enables the use of computational methods for analysis and interactive exploration (see Related Work for examples).
- ▶ Diagrams provide a formalized model that can serve as a scaffold for integrating *multi-omics* data.

Although their content is based on biological processes, disease maps differ in nature from other types of biological networks in the following aspects:

▶ The contents of a disease map are assembled based on the judgement of one or several curators. Only processes that are deemed relevant or informative to the given objective are included.

▶ A disease map is an actual visual diagram, i.e. visual representations of species and relationships have been laid out to optimally present the included information. Although several approaches for the drawing of large process diagrams exist (see TODO), it is still common practise to invest manual effort into the layout.

▶ Comprehensive disease maps can contain several thousands of visual elements. This poses special challenges particularly for layout and interactive exploration.

The above points also show that such maps are inherently subjective.

Recent disease maps contain up to several thousands of species and reactions and are very rich in information beyond the mere enumeration of species and their pairwise relationships. To give only a few examples: different types of species and relationships (as described above) are explicitly encoded. Further, species can be assigned relative positions to a biological compartment they reside in. Species can be assigned different states (e.g. "phosphorylated") and form *complexes* (groups). Further, species and relationships are often annotated with links to external databases such as Entrez Gene [6] or UniProt [7].

2.3 Drawing of Biological Process Diagrams

The most widely used and intuitive visualisation paradigm is the *node-link diagram* in which nodes are represented graphically as dots, circles or boxes and edges are represented by lines. For sake of simplicity, we use *nodes* and *edges* ambiguously both in their mathematical sense and for their graphical representations. Finding a *layout* of a graph (also called *graph drawing*) means finding positions for node representations and potentially also routing edges. What makes a good layout generally depends on the specific kind of network data and the task of the consumer of the visualisation. In general, a good layout is commonly required to avoid edge crossings or overlapping nodes, to be compact and to keep euclidean distances proportional to graph distances. Additional constraints may be for example the preservation of symmetries, clear representation of hierarchical structure or preservation of a mental map when updating a dynamically changing graph layout. Automatic graph layouting is a highly studied topic and numerous different approaches are available such as force-directed approaches [8], stress-based approaches [9], hierarchical or layered approaches [10] or orthogonal or grid-based drawings [11], to name just a few. Biological process diagrams, however, often contain subgraphs with particular semantics, which should be explicitly considered in the layout. Because of this, it is still common practise that biological process diagrams are drawn manually or are curated manually based on an initial automatic layout. This process is extremely time-consuming and often requires specific domain knowledge. Recent work [12][13][14] aims to capture these requirements and provide algorithms that produce a "human-like" layout (see Related Work). Some of the challenges in drawing biological process diagrams are [13] [12] [15] :

▶ Biological pathways often involve reaction cascades. These should be played in a way s.t. that the cascade is distuingishable and easy to follow. It may be preferrable to align cascades to the natural reading direction of the viewer (commonly top-to-bottom and left-to-right).

▶ Some biological pathways involve cyclic patterns and these should be clearly distuingishable as such.

▶ Process diagrams may be created to convey or highlight particular information and express an intentional message.

▶ Large diagrams such as the disease maps considered in this work often exhibit a clear structural and visual hierarchy.

▶ Biological process description diagrams may contain more information than merely species and reactions. Species can be (recursively) grouped into complexes. A complex is commonly represented as a box containing the visual representation of the species and thus its contents also need to be laid out. Further, cellular compartments are also commonly represented visually as large boxes or frames

containing the biological processes therein. Since transport in and out of the compartment and other reactions involving the membranes are of high biological relevance, proper placement of nodes and edges inside, outside or on the boundary of compartments is critical.

▶ Biochemical reactions often involve main substrates and products as well as secondary cofactors and enzymes. Substrates and products are usually placed orthogonally on opposite sides while cofactors and enzymes are placed on the side.

▶ A species often is relevant for several, potentially unrelated, biological processes. In this case, two separate visual representations (*duplicates*) are introduced in order to avoid displaying a false connectivity.

The question of when to duplicate a species alias is exactly the focus of this work. While structural features such as an extremely large node degree immediately come to mind, such simple predictors often do not suffice to capture all cases (see Experiments & Results). Although some software offers tools to ease the process of introducing duplicates, the decision of which nodes to duplicate is still left to manual curation (see Related Work).

Related terms   Several other terms appear in the literature that are relevant for the problem at hand. A *currency metabolite* [16] is a metabolite that plays only a secondary role in most of the reactions it is involved in. This role may be to merely supply energy (i.e. act as "currency" in the metabolism) or act as some other form of catalyst. Commonly, currency metabolites appear in abundance in an organism, and often it is assumed that two reactions involving the same currency metabolite are not in fact linked stoichiometrically, i.e. linking these reactions via a common node would imply false connectivity. Prominent examples are molecules such as ATP or H2O. Currency metabolites are commonly duplicated. The notion of true connectivity is also reflected in the concept of *key connectors* [17]: because pathways in a metabolic network seldomly work in isolation, there necessarily will be connections between different pathways. Determining whether a bridging node between two pathways is indeed a key connector or merely describes false connectivity and should be split is the tricky part (rephrase).

As such, it is problematic to hinge the decision of duplication on the identitiy of the node alone, since the same species may be a connector in some cases and not in other cases. As elaborated above, we have to consider its neighbourhood.

Node duplication is a crucial step in the curation of large diagrams since false connectivity introduces visual clutter in the form of long-range edges which in turn are hard to route and potentially introduce many edge crossings. Further, false connectivity disturbs the semantics of the visualisation and hinders its interpretation. Motivated by this, we seek to implicitly capture the criteria for node duplication by training a machine learning model on a sequence of snapshots of a diagram during a reorganisation process performed by an expert. The model is trained to predict whether a node should be duplicated or not.

Moreover, finding the exact number of duplicates to introduce and how to connect them is not trivial and was commonly left to human curators. We provide a basic approach to answer this.

Further, it remains an open to question where to position the newly introduced duplicates in the layout. We leave this question to future work.

2.4 Biological Databases and Ontologies

Different research projects on the same organism or disease deal in principle with the same universal sets of biological entities and relationships. For instance, if two projects were to describe the metabolic network of *E. coli*, both are likely to mention the function of Adenosine Triphosphate (ATP), a basic molecule that appears in many metabolic reactions. Both projects use the same notion of ATP and its effects, yet they may describe it differently, e.g., by its full name, abbreviation or chemical formula. Knowledge transfer and -integration relies on the interpretation and matching of identifiers by experts. Again, we are striving towards a structured, formalised body of knowledge for representing information about physical entities such as ATP, but also on biological terms describing processes (e.g. "glycolysis"), localisation (e.g. "cytoplasm") or functions.

There are several publicly available databases that gather information on biological entities such as proteins (*UniProt* [7]), genes (*EntrezGene* [6]) or drugs (*DrugBank* [18]). These databases often gather basic information and related research about the entity.

Likewise, biological terms describing processes, functions or cellular localisation can be represented in an *ontology*, a (directed acyclic) graph of terms and their relationships. For instance, the terms "glycolysis" and "carbohydrate metabolic process" may be connected by a *involved-in* relationship. The Gene Ontology project [19] provides three distinct ontology graphs describing molecular functions, cellular component or biological processes. Each ontology graph loosely describes a relationship, however, links between hierarchies may also exist, for instance (DNA repair (process), *occurs-in*, mitochondrion (component)).

2.5 Graph Embedding

2.6 Graph Neural Networks

The terminology used is still developing. There are many ways in which neural networks may exploit graph structure *. In this text, we will refer to neural networks that make use of message-passing layers in the style of **??** as *Graph Neural Networks*. Although the naming is somewhat ambiguous, this is the commonly used keyword.

Our main motivation is to predict whether a species alias in a disease map should be duplicated or not. For each species alias, we extract or compute *features*, which we deem to be characteristic for the target label of the node (see 13). Additionally, we are given training data in the shape of one or several disease maps and a binary label for each species alias describing whether it was duplicated during manual curation.

We are working in the setting of *supervised learning*, which we briefly introduce in the following [22, 23] : We consider n observations $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ (also called *training data*) to be drawn from an (unknown) distribution P over $\mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y}$ is the label domain and $\mathcal{X}$ is the feature domain. For the scope of this work, we restrict ourselves to the *binary classification problem*, i.e. $\mathcal{Y} = \{0, 1\}$. We assume that $\mathcal{X} = \mathbb{R}^d$ and the number of dimensions d is large. For sake of brevity, let us denote the set of observed *ground-truth* labels as $\mathbf{Y} := \{y_i\}_{i=1}^n$ and the set of observed feature vectors as $\mathbf{X} := \{x_i\}_{i=1}^n$. $\mathbf{X}$ can also be seen as a matrix in $\mathbb{R}^{n \times d}$. We assume that labels are generated by an unknown function f such that $y_i = f(\mathbf{x}_i)$. Further, we are given a *loss function* $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that describes how different a prediction is from the true outcome. In general, we aim to find a (parameterised) function $\tilde{f}$ that minimises the *true risk* $\mathcal{R}(\tilde{f}) := \mathbb{E}_P \left[\mathcal{L}(y_i, \tilde{f}(\mathbf{x}_i)) \right]$, where $\mathbb{E}_P$ is the expected value over P. Since the original distribution P is unknown, we instead seek to minimise the *empirical risk* on the training data $\mathcal{R}_{\text{emp}}(\tilde{f}) := 1/n \sum_{i=1}^n \mathcal{L}(y_i, \tilde{f}(\mathbf{x}_i))$ and assume the empirical risk approximates the true risk. In the following, we call a concrete choice of $\tilde{f}$ a *model* or a *classifier*. $\tilde{f}$ is commonly parameterised by a set of *model parameters* θ.

Neural Networks   A possible family of models that can be employed for classification are *neural networks*. We introduce neural networks by example of one of its simples variants, the *multilayer perceptron*, or *fully-connected neural network* and refer to [21] for a comprehensive introduction — The most basic building block of a neural network is a single *neuron*, which is given as the composition of a linear transformation with a nonlinear *activation function* σ. Formally, for a single input vector $\mathbf{x}_i$, the output of a single neuron is given by $\sigma(\mathbf{x}_i^T \mathbf{w} + \mathbf{b}_i)$ where $\mathbf{w}$ are called the *weights* of the linear transformation and $\mathbf{b}$ is the *bias*. The stacking of multiple neurons, of which each takes $\mathbf{x}_i$ as input constitutes a *fully-connnected* or *dense layer* in a neural network. If inputs and weights are stacked in matrices $\mathbf{X}$ and $\mathbf{W}$, respectively, the output of a single layer is given by $\sigma(\mathbf{X}^T \mathbf{W} + \mathbf{b})$ where σ is applied element-wise. A neural network is obtained by stacking multiple layers sequentially such that the output of one layer is the input of the next layer. The output of an intermediate layer is often called

---

* For instance, we do not consider here the mechanism of *pooling* through which the input graph can be coarsened w.r.t to its latent features. [20] This is another idea that has a correspondence in CNNs [21]

latent or *hidden representation*, i.e. if $\mathbf{H}^{(i)}$ is the output of the i-th layer, then the output of the $i + 1$-th layer is given as

$$\mathbf{H}^{(i+1)} = \sigma((H^{(i)})^T \mathbf{W}^{(i+1)} + \mathbf{b}^{(i+1)}). \tag{2.1}$$

Note that the hidden representation computed of some input may be considered an alternate feature representation of this input. In this sense, neural networks can be thought to perform feature extraction. For classification, a common approach is to have one output neuron for each target class.

The layer weights $\Theta = \{\mathbf{W}^{(1)}, ..., \mathbf{W}^{(l)}\}$ are considered to be the model parameters θ in the sense of the supervised learning framework described above. We aim to find a set of parameters such that the empirical risk is minimised. This objective can be optimised by *Gradient Descent*: We iteratively adjust the weights $\mathbf{W}^{(i)}$ such that the model prediction is improved with respect to the empirical risk $\mathcal{R}_{\text{emp}}$ [†]. More precisely, we consider the partial derivative of $\mathcal{R}_{\text{emp}}$ with respect to each weight matrix, which can be determined via the chain rule, and iteratively update the weights for a maximum number of steps or until the risk no longer improves. We can additionally specify the step size by a given *learning rate* η. Formally, in each step, also called *epoch*, we set

$$\mathbf{W}^{(i)\prime} \leftarrow \mathbf{W}^{(i)} - \eta \frac{\partial \mathcal{R}_{\text{emp}}}{\partial \mathbf{W}^{(i)}}.$$

Characteristics such as the number of layers, the number of neurons in each layer, the choice of loss function and learning rate determine the *model architecture* and are sometimes called *hyperparameters*. The proper choice of hyperparameters is essential to the performance of the model and often determined empirically via search techniques.

Automatic Feature Extraction and Message-Passing   In practise, selecting suitable features $\mathbf{X}$ is in itself a substantial step in the process of finding a well-performing classifier. However, particularly in high-dimensional domains $\mathcal{X}$, feature selection is not trivial. Consider the use-case of trying to find a model that, given a grayscale pixel image of dimensions (w, h), classifies the image based on whether it depicts a dog or a cat. It is not trivial how to manually extract features (criteria) from the image that are predictive of whether it depicts a dog or a cat. A straightforward approach would be to consider the brightness value of each input pixel as a feature, i.e. $\mathbf{x}_i \in \mathbb{R}^{w \times h}$. However, intuitively, looking at each pixel in isolation is unlikely to yield us a good classification since certainly the ground-truth classification depends on the values of pixels in their context.

Convolutional Neural Networks   For sake of intuitive appeal, let us further entertain the example of classifying pixel graphics. However, we emphasize that analogous formulations can be used for other kinds of structured data, including 1-dimensional grids (sequences) such as RNA and DNA sequence alignments [24][25]. It is important to note that we have additional information on the structure of the input namely that its pixel values are arranged in a grid. This means there is a well-defined notion of context, or *neighbourhood*[‡]. Aggregating information across a local neighbourhood in $\mathbf{X}$ may enable a model to capture not only characteristics of individual pixels but higher-order patterns. To faithfully extract local patterns, the aggregation operation should be *local* (the aggregation considers only a part of the input image) and *translation invariant* (the aggregation should respond similarly to the same input patch, regardless to where it is positioned in the entire grid). In the context of neural networks, such an aggregation is called a *convolution* [21]. In the case of linear aggregation and grid-structured data, we can express a convolution operation on pixel $\mathbf{H}_{i,j}$ as

$$\mathbf{H}_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} \mathbf{V}_{a,b} \mathbf{X}_{i+a,j+b} \tag{2.2}$$

[†] In the context of neural networks, $\mathcal{R}_{\text{emp}}$ is also sometimes called the total *loss*. This should not be confused with the loss function $\mathcal{L}$.

[‡] For sake of intuitive appeal, we give the following definitions for 2-dimensional grids but emphasize that

where u is the bias, and Δ is the (window) *size* of the *convolution kernel* **V**. **??** describes a *convolutional layer* and a neural network containing such layers is called a *convolutional neural network* (CNN).

Successive application of convolutional layers, potentially with different kernels can be thought to aggregate increasingly higher-order patterns in the input data. The extracted patterns can be considered intermediate, higher-order feature representations and the successive convolution operations extract increasingly higher-order features.

Note that the kernel can be any computation adhering to the constraints of locality and translation invariance. Since a kernel computes a value for a given pixel based on the features of itself and its neighbours, we can interpret the operation of a kernel to perform *message-passing*: Neighbour nodes construct and transmit messages to the target node, which are then combined with the target node's features into the final output.

Convolutions on graphs   We can extend the notion of a convolution quite readily from grid-structured graphs to arbitrary graphs. We will see that the convolutions considered here are trivially local and translation invariant. The key differences are that the order and the size of the neighbourhood is no longer fixed. As such, a potential aggregation must be *node-order equivariant*.

Let us consider an attributed graph $G = (V, E)$. Let $\mathbf{h}_i$ be the (intermediate) feature representation of vertex $v_i \in V$. Let $\mathcal{N}(v)$ be some neighbourhood of $v \in V$. Typically, $\mathcal{N}(v)$ is chosen as the 1-hop adjacency in G. However, note that different notions of neighbourhood can also be applied, so the input graph must not necessarily correspond directly to the computation graph that defines how messages are passed [§]. We can describe a simple convolution operation on attributed graphs as follows: the new latent representation $\mathbf{h}'_i$ of v_i is based on messages received by its neighbours. Each neighbour encodes its features by a message-passing function Msg. These messages are aggregated using a permutation-invariant aggregation function Agg. Finally, the neighbours' input and the node's own features $\mathbf{h}_i$ are coalesced via an update function Update into the new latent representation $\mathbf{h}'_i$. To summarise:

$$\mathbf{h}'_i \leftarrow \textsc{Update}(\mathbf{msg}_{ii}, \textsc{Agg}(\{\textsc{Msg}(\mathbf{h}_j, \mathbf{h}_i) \mid j \in \mathcal{N}_i\})) \tag{2.3}$$

Concrete choices of Msg, Agg and Update give implementations of *graph convolution layers*. A neural network containing such layers is commonly called a *graph neural network* (GNN) or *graph convolutional network* (GCN).

In simple GNN models, Msg is a linear feature extraction and depends only on the sending node, i.e. $\textsc{Msg}(\mathbf{h}_j, \mathbf{h}_i) = \mathbf{W}\mathbf{h}_j =: \mathbf{msg}_j$. Update is the application of a non-linear activation function and Agg is given by

$$\textsc{Agg}(...) = \bigoplus_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{msg}_j$$

where α_{ij} is a coefficient determing the importance of $\mathbf{msg}_j$.

The framework of **??** produces a latent feature representation for each node. This is the basis for solving node-level tasks such as the classification of clustering of individual nodes. Another common use case is to make a prediction for the entire input graph, the most prominent example in context of life sciences being molecule graphs. In this case, we want to aggregate all node features to produce a single value describing the input graph. Such an aggregation is called a *pooling* layer. This can be done either by application of a simple permuation-invariant aggregation function such as minimum, maximum, etc. or by iteratively coarsening the graph together with its node-level feature representations [20].

Simple GNN   The *GCN layer* [¶] as proposed by Kipf and Welling [27] defines α_{ij} as a constant depending on the degrees of v_i and v_j namely $\alpha_{ij} := 1/\sqrt{d_i d_j}$.

[§] For instance, GraphSAGE [26] samples a fixed number of adjacent nodes.
[¶] The naming here is unfortunately ambiguous.

GNNs with Attention   The *Graph Attentional Layer* (GAT) [28] allows the importance score to be learnable, i.e. adjusted via backpropagation and gradient descend during network training. An attention mechanism A determines the importance e_{ji} of $\mathbf{msg}_{ji}$. If the neighbourhood $\mathcal{N}_i$ is defined as the 1-hop-neighbourhood in the input graph, this can also be interpreted as the importance of edge (v_i, v_j). In its prototypical formulation, A is a single-layer fully-connected neural network with learnable weights $\mathbf{a}$, as described in **??**. A receives both the feature representations of v_i and v_j as input, combined by concatenation:

$$e_{ij} := A(\mathbf{msg}_i, \mathbf{msg}_j) = \sigma_{\text{att}}(\mathbf{a}^T(\mathbf{msg}_i \| \mathbf{msg}_j)) \tag{2.4}$$

Importance scores e_{ij} are then normalised via the *softmax* function:

$$\alpha_{ij} := \text{softmax}_{j \in \mathcal{N}_i}(e_{ij}) := \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}i} \exp(e_{ik})}. \tag{2.5}$$

Attention is a technique that has previously been used successfully in other neural network architectures. In particular, Graph Attention Networks are analogous to the Transformer architecture [29] that has achieved great popularity particularly in areas of Natural Language Processing such as machine translation. For translation from a source language to a target language, text input is typically given as two sequences of tokens. The key idea behind the Transformer architecture in NLP is to apply an attention mechanism to assess the importance of other words in the source or target sequence with respect to the current word. Note that while the Transformer architecture relies heavily on the attention mechanism, it also directly implements the idea of message-passing based on known relationships between atomic inputs.

<div align="center">

(diag-surfaces) on paper

</div>

Figure 2.1: Including known structure into the ML model as predictive bias.

Neural Networks as Classifiers   Intermediate layers of neural networks can be interpreted to compute useful feature representations. One way to obtain a classification prediction from a Neural Network is to use these intermediate representations as an input to another off-the-shelf classifier (such as Support Vector Machines, for instance [30]). Another, more common approach, however, is to design the network such that the last layer contains one ouput neuron for each target class. Given some input $\mathbf{x}$, the output value of the i-th neuron expresses the estimated probability that $\mathbf{x}$ is of class c_i. To produce values in $[0, 1]$, the results of the final layer are normalised, e.g. with the *softmax* function. In order to train the neural network, we need a loss function $\mathcal{L}$ to assess the quality of the prediction (the empirical risk). We introduce the *Binary Cross Entropy* loss from the viewpoint of maximum likelihood estimation (based on [21]) but note that it can also be derived via the notion of cross-entropy. Generally speaking, we seek to estimate parameters Θ of an assumed probability distribution P given some observed data X. The method of *maximum-likelihood estimation* postulates that we should pick the parameters such that the observed data is most likely (i.e. occurs with highest probability) under P. The likelihood w.r.t parameters Θ is measured by a *likelihood function $L(\Theta)$*. Formally, we aim for

$$\hat{\Theta} = \text{argmax}_{\Theta} P(X \mid \Theta) = \text{argmax}_{\Theta} L(\Theta) \tag{2.6}$$

Because of connections to entropy and computational convenience, we instead consider the negative logarithm:

$$\hat{\Theta} = \text{argmin}_{\Theta} - \log L(\Theta) \tag{2.7}$$

Looking at the negative log-likelihood in more detail directly yields the *Binary Cross-Entropy loss*. For notational convenience, let $\mathcal{Y} = \{0, 1\}$. Let $\pi_i = P_{\Theta}(y_i = 1 \mid \mathbf{x}_i)$ be the estimated probability that $\mathbf{x}_i$ belongs to class y_i. Since we are dealing with binary classification, we have $P_{\Theta}(y_i = 0 \mid \mathbf{x}_i) = 1 - P_{\Theta}(y_i = 1 \mid \mathbf{x}_i)$. Under

the assumption that input samples are independent and identically distributed, we have

$$-\log L(\Theta) = -\log \prod_{i=1}^{n} (\pi_i)^{y_i} \cdot (1 - \pi_i)^{1-y_i} \tag{2.8}$$

$$= -\sum_{i=1}^{n} y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i) \tag{2.9}$$

$$= \sum_{(\mathbf{x},y)\in\mathcal{T}} \text{CE}(f(\mathbf{x}), y) \tag{2.10}$$

where CE is the cross entropy loss and $\mathcal{T}$ is the set of training data. In summary, minimising the cross entropy loss is equivalent to maximising the likelihood.

2.7 Support Vector Machines

Support Vector Machines (SVMs) are a family of supervised machine learning models typically used for binary classification. In this section, we present the basic derivation of SVMs with the motivation of providing intuition behind the *cost* hyperparameter and choice of kernel functions and their parameters. We refer to [31] for a more rigorous and thorough treatment. Support Vector Machines are linear classifiers. The basic idea is to find a hyperplane in the (possibly transformed) feature space $\mathcal{X}$ that best separates the training data based on its ground-truth class assignments.

Preliminaries For sake of notational convenience, let $\mathcal{Y} = \{-1, 1\}$. A *hyperplane* is an affine set of points $L := \{\mathbf{x} \mid \mathbf{x}^T \beta + \beta_0\}$. The signed distance from a point $\mathbf{x}$ to L is given by $d(x, L) := 1/\|\beta\|(\mathbf{x}^T \beta + \beta_0)$. Based on L, we can define a linear classifier

$$h_L(\mathbf{x}) = \text{sign}(\mathbf{x}^T \beta + \beta_0). \tag{2.11}$$

$\mathbf{X}$ is *linearly separable* if there exists a hyperplane L such that $h_L(\mathbf{x}_i) = y_i$ for all $\mathbf{x}_i \in \mathbf{X}$, L is then called a *separating* hyperplane, or *decision boundary*.

Linearly separable case For now, assume that $\mathbf{X}$ is linearly separable. We wish to find a suitable separating hyperplane. One possible approach would be to minimise the distance of misclassified points to the hyperplane. Doing so by gradient descent yields the *perceptron training algorithm*. However, we are not guaranteed a unique solution. Further, if $\mathbf{X}$ is not linearly separable, the algorithm will not converge at all. Another possible approach would be to search for a hyperplane that maximises the *margin* $\gamma(L)$ w.r.t $\mathbf{X}$, i.e. the distance from the hyperplane to the closest point:

$$\gamma(L) := \min_{x \in \mathbf{X}} 1/\|\beta\| \left| \mathbf{x}^T \beta + \beta_0 \right|. \tag{2.12}$$

In maximum-margin separating hyperplane is unique. Further, it seems reasonable to assume that a maximum-margin separating hyperplane will do well in generalising to unseen points. We aim to find a hyperplane $L(\beta, \beta_0)$ that achieves:

$$\begin{aligned}
\underset{\beta, \beta_0}{\text{maximize}} \quad & \gamma(\beta, \beta_0) \\
\text{subject to} \quad & y_i(\mathbf{x}_i^T \beta + \beta_0) \geq 0; i = 1, \ldots, n
\end{aligned} \tag{2.13}$$

Since L and γ are scale invariant, i.e. $\gamma(\beta, \beta_0) = \gamma(\lambda\beta, \lambda\beta_0)$ for any $\lambda \neq 0$, we can assume $\gamma(\beta, \beta_0) = 1/\|\beta\|$, i.e. $\min_{x \in X} |\mathbf{x}^T \beta + \beta_0| = 1$. **??** is then equivalent to

$$\begin{aligned}
&\underset{\beta, \beta_0}{\text{maximize}} && 1/\|\beta\| \\
&\text{subject to} && y_i(\mathbf{x}_i^T \beta + \beta_0) \geq 0; i = 1, \ldots, m, \\
&&& \min_{x \in X}\left|\mathbf{x}^T \beta + \beta_0\right| = 1
\end{aligned} \tag{2.14}$$

and can simplified further to

$$\begin{aligned}
&\underset{\beta, \beta_0}{\text{minimize}} && \|\beta\| \\
&\text{subject to} && y_i(\mathbf{x}_i^T \beta + \beta_0) \geq 1; i = 1, \ldots, n
\end{aligned} \tag{2.15}$$

General case Let us now consider the case that **X** is not linearly separable. In this case, there is no solution to the optimisation problems given above. However, we can aim to relax the constraints so that some data points are allowed to lie inside the margin. We achieve this by introducing *slack variables* $(\xi_1, \ldots, \xi_n)$ to the optimisation constraints that express the allowed violation.

$$\begin{aligned}
&\underset{\beta, \beta_0}{\text{minimize}} && \|\beta\| \\
&\text{subject to} && y_i(\mathbf{x}_i^T \beta + \beta_0) \geq 1 - \xi_i, \\
&&& \xi_i > 0, \\
&&& \sum_{i=1}^{n} \xi_i = K
\end{aligned} \tag{2.16}$$

where K is some constant.

This optimisation is effectively solved by first transforming it into an equivalent problem known as its *Langrangian Dual*. To this end, it is convenient to express **??** as

$$\begin{aligned}
&\underset{\beta, \beta_0}{\text{minimize}} && 1/2\|\beta\|^2 + C \sum_{i=1}^{n} \xi_i \\
&\text{subject to} && y_i(\mathbf{x}_i^T \beta + \beta_0) \geq 1 - \xi_i, \\
&&& \xi_i > 0
\end{aligned} \tag{2.17}$$

The *cost* hyperparmeter C can be interpreted as a tradeoff coefficient between the cost of margin violations and simplicity of the decision boundary. For large C, margin violations will be punished more strictly. For small C, violations may be allowed to achieve a hyperplane with lower norm, i.e. smoother decision boundary.

The Kernel Trick For most non-trivial classification problems, the classification function f we seek to approximate is not linear. A method to move beyond linearity but nevertheless use linear classifiers is to transform the input features **X** into a higher-dimensional space $\mathcal{X}'$ via some transformation Φ. Depending on the choice of Φ, $\mathcal{X}'$ may be of very high or even infinite dimensionality. Thus, computing Φ explicitly is sometimes not an option. Luckily, we will see that in case of Support Vector Machines, all we need is an inner product $\langle \cdot, \cdot \rangle$ in $\mathcal{X}'$. In the following, instead of some feature vector $\mathbf{x} \in \mathcal{X}$, we consider a transformed feature vector $\Phi(\mathbf{x}) \in \mathcal{X}'$. Let's inspect the most central equations for computing SVMs. First, consider the

Lagrangian Dual of **??** given by

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle, \Phi(\mathbf{x}_i) \rangle \Phi(\mathbf{x}_j)$$

$$\text{where} \quad \beta = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$0 = \sum_{i=1}^{n} \alpha_i y_i$$

$$\alpha_i = C - \mu_i$$

$$\forall i: \quad \alpha_i, \mu_i, \xi_i \geq 0$$

Further, the decision function h_L (see **??**) can be rewritten as

$$h_L(\Phi(\mathbf{x})) = \Phi(\mathbf{x})^T \beta + \beta_0$$

$$= \Phi(\mathbf{x})^T \left[\sum_{i=1}^{n} \alpha_i y_i \Phi(\mathbf{x}_i) \right] + \beta_0$$

$$= \sum_{i=1}^{n} \alpha_i y_i \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}_i) \rangle + \beta_0$$

The key point here to note is that Φ appears only in the context of inner products. This means we can avoid even specifying Φ explicitly and instead use a *kernel function K* that expresses the inner products in the transformed feature space.

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$$

K is a valid kernel function if its Gram matrix is positive semidefinite. One popular choice is the *radial basis function* (RBF) kernel[‖], also called *Gaussian* kernel. Recall that a kernel function can be interpreted as a measure of similarity between its two arguments $\mathbf{x}$ and $\mathbf{x}'$. The RBF kernel implements this notion as a decaying function of their distance. If $\mathbf{x}$ and $\mathbf{x}'$ are similar their distance will be small, $-\gamma \|\mathbf{x} - \mathbf{x}'\|^2$ will be relatively large. The decaying characteristic is implemented by the application of the exponential function. Thus, we can define the RBF kernel as

$$K_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2). \tag{2.18}$$

Since the distance is symmetric, K_{RBF} can be interpreted as a bell-shaped function. The hyperparameter γ controls the width of the bell shape, with larger values producing a more narrow shape.

[‖] This is in fact an example where the transformed feature space is of infinite dimension: It can be seen via Taylor expansion that K_{RBF} is based on an infinite sum of polynomial kernels.

3 Related Work

Of particular prominence is the work of Nielsen et al. [32].

3.1 Drawing Biological Networks

Tools   Several tools exist for the curation and exploration of biological process diagrams, including *CellDesigner* [33], *Minerva* [34], and *Cytoscape* [35] and *VANTED* [36]. We refer to TODO for a comprehensive comparison.

3.2 Node Duplication

General methods   While node duplication (also referred to as *Vertex Splitting*) has been treated from a theoretical perspective [37] [38], to the best of our knowledge there are relatively few concrete, general-purpose algorithm that employ automatic node duplication.

Most notably, Eades and de Mendonça N introduce a general graph drawing algorithm that applies node duplication to simplify the graph structure in order to find a better layout [39]. The method is an extension of the force-directed Kamada-Kawai algorithm [40]. Given a vertex v and an incident edge, they define a *tension* vector whose direction is the direction of the edge in the current drawing and whose magnitude is based on the difference between the actual edge length in the current drawing and some target distance. Any possible binary duplication defines a *split line* that partitions $\mathcal{N}(v)$ into two disjoint subsets $\mathcal{N}(v)^+$ and $\mathcal{N}(v)^-$. The *tension of a split line* is the sum of tensions of either $\mathcal{N}(v)^+$ or $\mathcal{N}(v)^-$ (they are in fact equal if the drawing is in equilibrium). The *tension of a vertex* then is the maximum tension of all possible split lines. In each iteration of the Kamada-Kawai algorithm, additionally, a vertex with tension greater than a given threshold is split into two duplicates according to its maximum tension split line. The choice of the threshold is left to the user.

Node Duplication has also been applied in the area of Electronic Circuit Design, for example to avoid edge crossings [41] or paths exceeding a maximum length [42] [43].

Henr, Bezerianos, and Fekete consider how to represent duplicates in a social network visualisation in which communities are represented by adjacency heatmaps [44].

Metabolic Models   Because different instances of the same metabolite can participate in many different reactions, node duplication can be essential for working with metabolic models. While visualisation one possible motivation, node duplication also has to be considered for other tasks such as flux balance analysis or integration of secondary *-omics* data [45].

Node duplication in layout tasks potentially is a different problem from the general preprocessing since finding a good layout may come with particular constraints (such as considering edge crossings, stress etc.). However, the methods mentioned below do not take layout information into account when deciding node duplication and we thus make no explicit distinction here.

Since nodes and particular motifs carry semantic biological meaning, deciding node duplication in metabolic models poses additional challenges.

Any node v with degree greater than two introduces a connectivity between any two subsets of $\mathcal{N}(v)$. It is not trivial to distuingish whether a connectivity is *false*, i.e. only exists because participants in two reactions happen to be mapped to the same concrete node or *true* in the sense that the connectivity represents actually meaningful biological information.

Review   A common characterisation is that a node shall be duplicated if its neighbourhood is highly heterogenous w.r.t some measure. In other words, a node should be split if it is involved in many different, unrelated processes as then it is assumed that in reality there are independent instances of that species, involved in the different processes independently.

One possible approach is to consider a graph-based centrality measure. If the target node has a very high centrality score, we make the assumption that then the node must necessarily be involved in different, unrelated processes, i.e. its neighbourhood is heterogeneous. For a concrete choice of centrality measure, early methods simply considered the node degree [46] [47]. Further work uses the Eigenvector centrality [45].

Other approaches make the notion of neighbourhood heterogeneity more explicit by characterising communities in the given network. A node then has heterogeneous neighbourhood, if it is incident to many different communities. Communities can be determined solely on the graph structure, for instance as induced by modularity maximisation [48]. Huss and Holme decide node duplication based on the contribution to overall modularity if the target node is removed [16]. Guimerà and Nunes Amaral classify nodes as different kinds of hub- or connector nodes based on their intra- and inter-community degrees, where communities are determined via modularity maximisaiton [49]. Communities can also be characterised by domain-specific biological knowledge, i.e. annotations that describe the cellular compartment [45] or the pathway [50] [51] for a given node.

Apart from the work of Nielsen et al., we are not aware of any other previous work that investigates using a classifier for node duplication.

Tools and Formats   There are several tools that provide functionality to easily introduce duplicates once the decision has been made. *Arcadia* [52] allows to split a node such that each copy has exactly unit degree. *Omix* [53] makes it easier to introduce duplicates with certain connectivity patterns by providing a *motif stamp* tool.

One of the most common formats used for describing disease maps is an extension to SBML Level 2 given by *CellDesigner*. Further, SBML Level 3 now allows to attach layout information. Another prominent format is SBGN-ML.

3.3 Disease Maps

To date, disease maps have been created for a number of diseases, including Alzheimer's Disease (AlzPathway[54]), Parkinson's Disease (PDMap[55]), and recently COVID-19 [56]. Beyond serving as a platform for integrating existing knowledge, computational methods have been applied to, e.g., identify molecules and relations essential for the pathogenesis of Alzheimer's Disease [57].

[58]

Detailed information on the disease maps considered in this work can be found in Section 4.1.

3.4 Machine Learning on Biological Networks

The life sciences offer countless applications for machine learning methods. Herein, we constrain ourselves to methods that directly or indirectly work with network-structured data and apply neural network techniques, or are particularly relevant for this work. Further, for sake of brevity, we omit edge-level tasks such as link prediction in biological networks.

Applications of Graph Neural Networks

Applications of GNNs on biological are diverse in the formulation of the task, the underlying data (network and attributes) and the applied methods.

To the best of our knowledge, there is very little previous work that connects GNNs to graph drawing or even node duplication.

Tiezzi, Ciravegna, and Gori provide a method for layouting graphs using GNNs [59] . As a first approach, they train a GNN to draw graphs based on ground-truth examples obtained from other graph drawing software. As attributes, each node is assigned a positional encoding based on the Laplacian Eigenvectors of the graph. The loss function aims to minimise the distances from the produced drawing to the ground-truth drawing (modulo affine transformations). Further, inspired by optimisation-based methods such as Stress Majorisation [9], they employ a GNN to directly minimise the stress function on the predicted node coordinates. Note that the all-pairs-shortest-paths computation has to be done only during training. At inference time, the model predicts node positions based on the supplied positional encoding, which is potentially easier to compute. Additional quality measures, such as the number of edge crossings, can be included in the loss function without sacrificing the advantage that predictions only ever require the graph structure and positional encodings and no additional computation.

Node-level tasks   You et al. [60] consider the problem of predicting the function of a protein based on its sequence. A current challenge is bioinformatics is the gap between the number of known protein sequences and the number of protein sequences annotated with a biological function (*sequence-annotation-gap*). You et al. suggest to consider each protein with respect to other proteins it is known to interact with. As such, they propose a GNN approach in which proteins are represented as nodes and connected based on information from Protein-Protein interaction databases. The function annotations here are in fact Gene Ontology terms (see TODO). Once the biological function is known, another challenge is to assess the functional similarity of two proteins. One way to approach this is to compare the (sets of) Gene Ontology terms of two given proteins. Zhong, Kaalia, and Rajapakse compute an embedding for GO terms based on the graph structure of the GO ontology [61]. These embeddings can be thought of to encode the term's position in the graph, that is, its relationship to other terms. A measure of semantic similarity can be derived by comparing the (sets of) computed embeddings.

Single-cell RNA-sequencing (scRNA-seq) is a technology that provides gene expression data on the level of individual cells. In the work of Ravindra et al. [62], each cell is represented as a node and features are its gene expression data. Graph connectivity is defined on a node's k nearest neighbours. The goal is to predict whether a cell corresponds to a healthy or pathological disease state w.r.t to Multiple Sclerosis (MS). The motivation is to work towards developing a diagnostic test for MS based on scRNA-seq technology.

Graph-level tasks   Graph Neural Networks have found highly successful applications for the problem of molecular property prediction. This is a graph-level task where the input is a *molecule graph* and we strive to predict specific chemical properties of this molecule. In a molecule graph, nodes represent atoms and edges represent bonds. Nodes and edges have attributes describing e.g. an atoms element, or the type of a bond. Properties to predict may include toxicity or antibacterial activity. This task is relevant particularly in the field of drug development where a large number of molecules has to be tested for potential usefulness as a drug (*virtual screening*). Traditionally, molecules were represented by *fingerprint vectors* which describe characteristics of the entire molecule such as the presence of functional groups. While these fingerprints may well serve as input to a neural network predictor, the structure of these fingerprints is designed manually and often not directly dependent on the prediction task. Graph Neural Networks provide the means to compute such fingerprints directly based on the the structure of the molecule and concrete, low level properties. Moreover, the extraction and aggregation of input features performed by graph neural networks is differentiable and thus the entire prediction pipeline can jointly be optimised end-to-end, yielding task-specific fingerprints. Stokes et al. use this approach to predict the growth inhibition against *E.coli*, eventually resulting in the discovery of experimentally verified potent antibiotics that are structurally different from known

antibiotics [63]. Notably, Duvenaud et al. employ this approach well before the recent popularisation of Graph Neural Networks in the style of **??**. Focussed reviews of applications of GNNs for molecular property prediction [65] and drug development in general [66] can be found in the literature. Further, Baranwal et al. train a GNN model to compute fingerprints of molecules that are then used to predict their broad metabolic pathway class (e.g. carbohydrate metabolism, amino acid metabolism, *etc.*) [67].

Edge-level tasks   GNNs have been applied for edge-level tasks particularly for the problem of link prediction in biological interaction networks. GNNs have been applied to predict interactions between diseases and drugs [68], interactions between drugs, proteins and drug side effects [69] and interactions between proteins [70]. Many other applications can be found in the literature [71].

Other relevant ML approaches

3.5 Semantic Representations

Gene Ontology

[72] [73]

Work exists to develop similarity measures between two GO terms, or two sets of GO terms (see Section 3). These are, by definition, pairwise similarity measures. These could be interpreted as distance measures and these distances could be considered directly in a machine learning model.

An alternative approach is to find a embeddings (vectors describing feature representations) of the given GO term s.t. the similarity of their embeddings is meaningful, i.e. reflects the similarities in the GO graph. In fact, some approaches for semantic similarity measures depend exactly on such embeddings.

### Embeddings

4 Methods

4.1 Datasets & Preprocessing

Datasets used for training and evaluation

A visual overview of the disease maps considered in this work is provided in Figure **??**.

The *AlzPathway* map describes signaling pathways related to Alzheimer's Disease. Since its initial publication [74], it has received several updates and further analyses [75] [54] [57]. The map has received additional curation focussing on increasing readability by means of reorganising existing network elements. [76]. This includes the duplication of some species aliases. During curation, snapshots of intermediate progress (*reorganisation steps*) were saved. Note that the reorganisation steps are not atomic: each step includes modifications to several nodes and edges. This sequence of reorganisation steps served as the basis for the work of Nielsen et al. to train an SVM classifier to predict node duplication [32]. In this work, we consider these reorganisaton steps for training data (ALZPATHWAYREORG). Further, we consider the last of the reorganisation steps, i.e. the final result as a single, independent map (ALZPATHWAYREORGLAST).

The *PDMap* [55] describes the major pathways involved in the pathogenesis of Parkinson's Disease. The map can be explored via a hosted *Minerva* instance, reachable at `https://pdmap.uni.lu/minerva`. For this work, we consider the version of this map as exported from *Minerva* at 2021-09-07.

ReconMap [77] is a visual representation of the genome-scale metabolic model *Recon 2* [78] that aims to comprehensively model the human metabolism. It can be viewed at `https://vmh.life/minerva`.
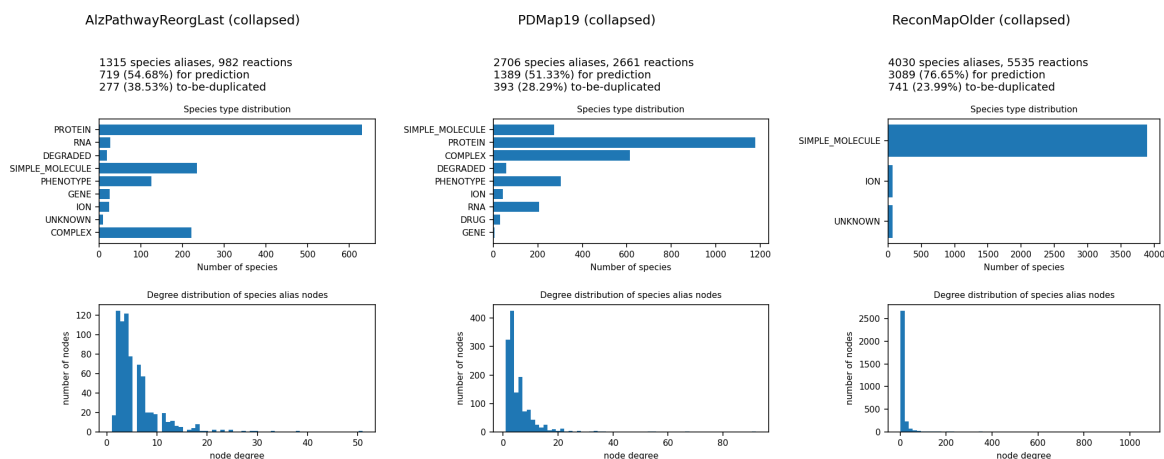
Figure 4.1: An overview of characteristics of the *collapsed* networks used for training. The count of species aliases and reactions is effectively the count of the bipartite node sets in the constructed graphs. Since we are consiering collapsed diagrams, the count of species aliases equals the count of species. Networks and labels are determined as described in Section 4.1.

Graph construction

Disease maps can be interpreted as bipartite graphs in a natural manner with the bipartite node sets being the set of reactions and the set of species, respectively. The CD-SBML format is very rich in information and leaves some room for ambiguity concerning graph construction. In the following, we describe what we take into account to construct a graph.

The most central elements in an SBML model are the lists of reactions and species aliases. The basic idea is to construct a bipartite graph in which species aliases and reactions form a bipartite node set.

An entry in the list of reactions carries references to species taking part in that reaction. Since different occurences of a species can be visualised as duplicate species aliases, each entry for a participating species additionally contains a reference to a specific species alias. Participating species are distuinguished by the role they play in that reaction. Herein, we consider the basic roles defined by standard SBML: products, modifiers and reactants *. We create directed edges for reactants and products in the direction of the reaction. A modifier is attached by two edges, one in either direction. For computing structural node features and for message-passing, we sometimes also consider the bipartite projection. Herein, we compute the bipartite projection based on the undirected interpretation of the graph.

(graph-interpretation) from paper notes

(a) Illustration of how a rich SBML model is interpreted as a simple, directed graph.

todo: something illustrating CD-SBML structure / distinction between species and species aliases – or not at all? not really super relevant...

A species alias can either be simple or *complex* in the sense that it represents a container for other species aliases. This is used to represent e.g. biological complexes of proteins. Simple and complex species aliases are arranged in an arbitrarily nested hierarchy. For the graph structure, we consider only top-level elements, that is, we consider complex species aliases as single nodes and omit their contents †. A reaction may involve an entire complex species alias (CSA) or only a species alias contained in the CSA. Since we interpret the CSA as a single node, an edge will be attached to it in either case.

Species aliases can also be contained in *compartments* representing biological cellular compartments or broader notions of spatial relationship. We do not consider compartments at all herein (see Section 7).

Determining ground-truth Labels

Although numerous disease maps are publicly available, to the best of our knowledge none are explicitly annotated with a per-alias label indicating node duplication. In case we are given a sequence of reorganisation steps $(G_1, ..., G_k)$, we infer node labels by comparing successive steps G_t and G_{t+1}. In case we are given only a single disease map G, we first construct a collapsed version G_0 by collapsing any species aliases corresponding to the same species into a representative node and moving any edges incident to aliases to the corresponding representative. We then proceed by comparing G_0 and G like reorganisation steps. In order to make our results comparable to the work of Nielsen et al. [32], we employ the same algorithm for inferring ground-truth labels. Because the algorithm has received little explicit treatment in the original publication, we motivate and describe it here in detail for clarity.

A simple approach that comes to mind is to consider nodes newly introduced in G_{t+1} and look for a subset $W \subset V(G_{t+1})$ whose neighbourhood completely covers the neighbourhood of some node in G_t, i.e. $\bigcup_{w \in W} \mathcal{N}_{t+1}(w) = \mathcal{N}_t(v)$ for some $v \in G_{t+1}$. However, this does not suffice. It is important to note that we can make no assumptions about what manipulations were made to create G_{t+1} from G_t. In particular, nodes may have been removed, added, duplicated (in the true sense), copies introduced and edges may have been added, removed or re-wired. To work around this, instead of finding the duplicates of a given node in G_t, the algorithm seeks to identify a possible *duplication parent* of a given node in G_{t+1}. The basic idea is that the neighbourhood of duplicates will be at least partially included in the neighbourhood of the duplication parent. We aim to find duplication parents by starting at $v_i \in G_{t+1}$, taking one step away in G_{t+1} and then taking one step back again in G_t.

The procedure is given in pseudocode in **??**. For directed graphs, let the *positive neighbourhood* of v in G_k $\mathcal{N}_k^+(v)$ be given as $\{w \mid (v, w) \in E(G_k)\}$ an the *negative neighbourhood* $\mathcal{N}_k^-(v)$ as $\{w \mid (w, v) \in E(G_k)\}$. The set

---

* *CellDesigner* provides an even more fine-grained distinction of species participating in a reaction. Species can be *main* reactants or products, *modifiers* or *additional* reactants or products. We omit this for simplicity. See [79], ch. 2.4.
† Contents of complex species aliases will be taken into account when considering GO annotations, see TODO

Algorithm 1: Procedure to identify duplication parents given a reorganisation step. Transcribed from Nielsen et al.[32].

Data: Directed graphs G_t and G_{t+1}, node $v_i \in G_{t+1}$

Result: Duplication parent of v_i or None

1 $W_+ \leftarrow \mathcal{N}_t^-(\mathcal{N}_{t+1}^+(v_i))$

2 $P_+ \leftarrow \mathcal{N}_{t+1}^-(\mathcal{N}_{t+1}^+(v_i)) \setminus W_+$

3 $W_- \leftarrow \mathcal{N}_t^+(\mathcal{N}_{t+1}^-(v_i))$

4 $P_- \leftarrow \mathcal{N}_{t+1}^+(\mathcal{N}_{t+1}^-(v_i)) \setminus W_-$

5 **if** $W_+ = \emptyset \vee W_- = \emptyset$ **then**

6 $P \leftarrow P_+ \cup P_-$

7 **else**

8 $P \leftarrow P_+ \cap P_-$

9 **if** $\|P\| = 1$ **then**

10 Let w be the single element in P

11 **return** w

12 **else**

13 **return** None

operations identify nodes solely based on their alias ID. This means that $V(G_t)$ and $V(G_{t+1})$ may potentially have nonempty intersection.

Line 8 states that a valid duplication parent must be reachable from both positive and negative direction. This is only relevant if there is no positive (resp. negative) neighbourhood shared between the reorganisation steps (which is also the case if the target node is a sink, resp. source). Line 6 takes this into account. Then, a duplication parent may still be uniquely identified if there is a single shared neighbour in the opposite direction (see Figure **??**). Line 9 handles cases when multiple candidate duplication parents exist and we cannot infer a unique single one (see example Figure **??**). The algorithm is able to identify duplication parents even if edges have been removed (see example Figure **??**).

subfig

(a) Example in which there is no shared positive neighbourhood. However, a unique duplication parent can still be identified because node 2 has unit out-degree.

subfig

(b) Example where a unique duplication parent cannot be identified due to ambiguity.

subfig

(c) Parents can still be inferred if edges are removed. Without node 7, however, the case would be ambiguous

subfig

(d) Ambiguities are resolved by considering both positive and negative neighbourhoods.

Figure 4.3: Examples for Algorithm **??**

Feature Engineering

We consider the following attributes as node features. We investigate which subset of these features serves well as predictors in Section 5.

In general, there are three aspects of a disease maps based on which features can be defined. The first is the *structural* aspect in which we use graph-theoretical measures to characterise species aliases (nodes) based on their connectivity in the network. This was considered in detail in the feature engineering step in the work of Nielsen et al. and was used in numerous approaches for metabolic networks (see Section 3). The second aspect is *semantic*: species are annotated with biological domain knowledge, commonly in the form of links to databases that provide additional information about that species (see TODO background). In this work, we aim to explore how to exploit Gene Ontology term annotations for node duplication. The

third aspect is that of *layout*: Prior to making a prediction on node duplication, the disease map may already have been laid out to some extent. If so, positions of species aliases certainly carry some meaning. Which exactly, however, is unclear since positions are most likely determined by a mixture of layout requirements, semantic arrangement and preference of the curator. For the scope of this work, we avoid using layout-based predictors since we only have one practical instance in which we have layout information and ground-truth labels available, which is when making a prediction for a AlzPathway reorganisation step and comparing it to the actual next step in the reorganisation sequence. However, this is the only disease map for which reorganisation steps are available and it is of limited size. Thus we decide to omit this approach for now, while acknowledging that considering layout information may lead to interesting approaches (see Section **??**) [‡] .

Structural Features   Because the input networks are of nontrivial size (see refsec datasets used), computation time has to be taken into account. Particularly attributes relying on global information (such as e.g. closeness centrality) can provide a challenge for high-level graph libraries such as *networkx* and may not complete in a reasonable amount of time. More sophisticated implementations working with a lower-level language such as those provided by the *igraph* library used herein show a significant speedup.

The following structural features were defined and used by Nielsen et al. and we re-implement them for comparability. Let $\sigma_{st}(v)$ denote the number of shortest paths from s to t passing through v and σ_{st} is the number of shortest paths from s to t. Since in the following, we always consider the graph to be undirected, σ is symmetric w.r.t s, t. Let $\mathbf{A}$ denote the binary, symmtric adjacency matrix.

- ▶ degree: The degree of a node, counting both incoming and outgoing edges.
- ▶ clustering_coefficient: The *clustering coefficient* [80] for a node v is given as

$$\frac{2\tau(v)}{\deg(v)(\deg(v) - 1)}$$

  where $\tau(v)$ is the number of possible triangles through v in the graph.
- ▶ betweenness_centrality: The *betweenness centrality* of v reflects the number of shortest paths that pass through v:

$$\sum_{s \neq v} \sum_{t \neq v} \delta_{st}(v)$$
$$\text{where } \delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

- ▶ closeness_centrality: There are different ways to define a closeness-based centrality measure. The definition used here is:

$$\frac{\|V\| - 1}{\sum_{s \neq v \in V} d(s, v)}$$

  where $d(s, v)$ is the shortest-path distance from s to v.
- ▶ eigenvector centrality: The *eigenvector* centrality is a measure of transitive importance of a node. The basic idea is that a node is important if it is linked to other important nodes. A set of centrality values reflecting this notion is given by the eigenvector $\mathbf{x}$ of $\mathbf{A}$ corresponding to the largest eigenvalue.
- ▶ neighbour_centrality_statistics: For node v, this the stacked vector of statistics over several centrality measures of neighbours of v. The statistics are mean, minimum, maximum and standard deviation. The centralities are betweenness, closeness and eigenvector centrality and degree.
- ▶ distance_set_sizes: A vector of length k in which the k-th entry is the normalised number nodes exactly k hops from v. Here, $k = 5$ is considered. The values are normalised by the number of nodes reachable from a given node in a grid graph via k hops.

[‡] We do consider layout information as a criterion for attaching edges to duplicates, see Section 4.3.

Except for the clustering coefficient, additionally all characteristics are also computed on the bipartite projection and included as separate posible feature. Except for `distance_set_sizes`, all features are min-max-normalised w.r.t all given training graphs.

Additionally, we consider the directed degrees (`in_degree`, `out_degree`). This is motivated by the notion that a species alias may have a different biological meaning w.r.t node duplication if either in- our out-degree is higher, both are balanced or the node is a source or a sink.

Semantic Features   We hypothesize that, in general, a node should be duplicated if its neighbourhood is highly heterogeneous, i.e. the species alias establishes false connectivity between actually unrelated subgraphs. Previous approaches commonly focus on structural or layout features. However, these facets involve domain knowledge only implicitly at best. As a characterisation of neighbourhood homogeneity, we seek to explicitly include domain knowledge about what biological processes a species is involved in. To this end, we consider the Gene Ontology terms linked to each species in the given disease map. In particular, we consider those terms of the "*biological process*" subtree of the GO graph (see Section **??**).

We extract GO annotations directly from the given CD-SBML files.

For the purpose of this work, we chose the direct, embedding-based approach for the following reasons: We are alleviated of the choice of an additional similarity measure (effectively leaving that problem to the classifier). This also means that particularly in case of a GNN classifier, the neural network may potentially be able to capture more flexible relationships than what we would encode with a particular similarity measure. Next, this gives us a simple approach for including annotation information for complex species aliases by aggregating the embeddings of the contained nodes. Additionally, since in the end we aim to assess the heterogeneity of a node's neighbourhood, the consideration of embedding values could be incorporated in the message-passing step of a neural network (see Section 7). Further, node-level information may be useful for the task of finding an attachment of edges after duplication. For computing an embedding vector for a given GO term, we closely follow the approach of Zhong, Kaalia, and Rajapakse [61] and encode a term's position in the GO graph via the *node2vec* algorithm (see Section **??**).

There are two cases when we are required to aggregate a set of embeddings into a single value. On the one hand, a single protein can be annotated with several GO terms. On the other hand, a complex species alias potentially contains muliple annotated aliases. In both cases, we take the mean.

This results in the following two concrete feature definitions:

▸ `GO_embedding`: The basic idea of this approach is to only provide an encoding of the term's position in the GO graph as node feature. A GNN model could then potentially capture characteristics of a target node's neighbourhood via its aggregation function.

▸ `GO_stddev`: We additionally try capturing neighbourhood heterogeneity by interpreting the embeddings as points in an euclidean space and deriving a measure for the spread of these points around their centroid (mean). The idea is inspired by the measure of standard deviation, i.e. the average squared distance from the mean. This can conventiently be expressed as the sum of variances over each dimension. Let $(\mathbf{x}_1, ..., \mathbf{x}_n)$ be the embeddings of neighbour nodes, and let $\bar{\mathbf{x}}$ be the centroid (i.e. dimension-wise mean). Assume the points lie in a D-dimensional euclidean space. Let $(\mathbf{x}_i)_k$ denote the k-th entry of $\mathbf{x}_i$. We then consider $\sigma = \sqrt{\sigma^2}$ as a measure of spread, given by

$$\sigma^2 = 1/n \sum_{i=1}^{n} \|\bar{\mathbf{x}} - \mathbf{x}_i\|_2^2$$

$$= \sum_{d=1}^{D} 1/n \sum_{i=1}^{n} (\bar{\mathbf{x}})_d - (\bar{\mathbf{x}}_i)_d)^2$$

$$= \sum_{d=1}^{D} \mathrm{Var}\big([(\mathbf{x}_1)_d, ..., (\mathbf{x}_n)_d]\big)$$

where $\mathrm{Var}\left(\left[(\mathbf{x}_1)_d, ..., (\mathbf{x}_n)_d\right]\right)$ is the variance along dimension d. Note that this approach encodes characteristics of the embeddings across a node's neighbourhood directly into a single feature vector and can thus be used with any classifier, such as SVMs.

4.2 Training & Classification

The basic pipeline is illustrated in Figure 4.4. The general approach is to extract data from the CD-SBML files describing some disease maps, and constructing for each an attributed, directed, bipartite graph G. Additionally, we infer ground-truth labels for nodes in G. We aim to classify nodes in G. For each node, we compute a feature representation. For the SVM model, the feature vector and ground-truth label is the only input. For GNN models, additionally the network structure is given.

Nodes corresponding to complex species aliases and nodes of degree less than two are excluded from prediction. This means they will not be considered as input examples when training or evaluating the classifier. Note that these nodes are still part of G and potentially influence the features of other nodes. Further, excluded nodes will still participate in the message-passing steps of GNN models.

The set of thus constructed input graphs is partitioned into training and testing graphs. For the concrete choice of training and testing partitions, we explore different settings, see Section 5.

(diag-pipeline) on paper

Figure 4.4: Basic pipeline

The task at hand comes with a particular set of challenges:

▶ Different disease maps potentially have very different characteristics (see e.g. the difference between AlzPathwayReorg or PDMap and ReconMap in Figure **??**), which may make it difficult for the model to generalise to unseen disease maps. We evaluate generalisation ability in Section **??**.

▶ Typically only a few nodes are duplicated (see Figure **??**). Thus, the positive class is underrepresented in the dataset. We have to ensure that model performance does not suffer due to class imbalance. We address this issue by considering two approaches: First, we undersample the majority class. Second, we provide class-specific weights to the ML models such that an example of the minority class will be assigned more importance.

▶ The ground-truth labels may not be perfectly reliable. For one, the decision whether a node was duplicated or not during curation is subjective to the expertise and preference of the curator. The criteria that were relevant to the curator are most likely not perfectly reflected in the features we provide the ML model (TODO which is the entire motivation for a ML approach with very complex models). Thus, it may be the case that some training examples are contradictory in label w.r.t.their feature representation. Further, the procedure to infer ground-truth labels (see Section **??**) may not be perfect and indeed miss some reorganisations that could indeed be seen as duplication events.

▶ The number of datapoints used for training and evaluation is relatively low compared to other common use-cases for Machine Learning. Care has to be taken to avoid overfitting, i.e. the model adjusts overly well to the training data at the cost of prediction performance on validation data not seen during training. Further, particularly when partitioning the available data into subsets for training and testing, we have to make sure we still have plenty of representative examples in each subset. We address this issue in part by avoiding to split a disease map internally into training and testing subsets. Further, we want to highlight that SVMs an GNNs have been applied successfully on datasets of comparable size. Further, the GNN architectures we considere herein are of much smaller complexity (in terms of number of model paramaters) than famous neural network architectures used in Computer Vision or Natural Language Processing and may thus not require a particularly large amount of training data to fit all parameters.

4.3 Attachment of edges

Assume we are given a binary classifier that decides whether a node should be duplicated. If a node v is eligible for duplication, we need to determine how many duplicates to introduce and how to distribute edges to and from v across the duplicates. Formally, we aim to find a partition of $\mathcal{N}(v)$. Based on the intuition that a good duplication is one that reduces the heterogeneity of $\mathcal{N}(v)$, we can characterise the partitions as *clusters* in the sense that intra-cluster distances are smaller than inter-cluster distances.

The choice of distance metric is open. Of particular interest are metrics that reflect semantic similarity of attached GO terms [§]. However, because in the given datasets, most aliases are annotated with a relatively large number of GO terms and exploiting these annotations for classification did not yield gains in classification performance most likely due to high ambiguity, we opt for a simpler approach first and leave this open to future work. Instead, we consider the layout positions of aliases and their euclidean distances. Note that this approach is only applicable if layout information is actually given, i.e. this approach can not be used if we construct a collapsed graph from a single given disease map (unless we would infer layout information for the newly constructed, collapsed graph).

Particularly when considering euclidean distances in the layout, a suitable clustering algorithm needs to handle outliers in a sensible manner. Additionally, we do not know the number of clusters in advance. This eliminates basic partition-based methods such as k-Means and extensions. Further, we seek to assign all points to a cluster, i.e. we do not want to exclude any points as noise. Also, since different node neighbourhoods have potentially different scales, we aim to avoid having to specify hyperparameters like distance thresholds as they are used in density-based clustering algorithms like DBSCAN or Optics. A family of clustering algorithms that seems well suited is that of *agglomerative* clustering: Initially, each point is assigned its own cluster. Iteratively, the two clusters with the smallest inter-cluster-distance are merged until only a single cluster remains. This yiels a hierarchical clustering tree, also called *dendrogram*, as depicted in Figure **??**.

There are several choices of distance measures between two clusters C_1 and C_2. The most simple ones employed for agglomerative clustering are:

$$single\ linkage :\ d(C_1, C_2) = \min_{p \in C_1, q \in C_2} d(p, q)$$

$$complete\ linkage :\ d(C_1, C_2) = \max_{p \in C_1, q \in C_2} d(p, q)$$

$$centroid\ linkage :\ d(C_1, C_2) = d(\text{mean}(C_1), \text{mean}(C_2))$$

where C_1 and C_2 are considered to be sets of points. Complete linkage and centroid linkage seem inadequate since they would be strongly affected by large in-cluster variances and do not work well if clusters are not convex.

Setting a threshold value on the maximum dissimilarity inside a cluster, we obtain a concrete clustering. This can be thought of as "cutting off" the dendrogram at a specific height. Note that here we do not have to specify the number of clusters but instead a threshold dissimilarity. This threshold can be determined automatically via a heuristic [¶]: We look for the strongest increase in the distance to the next closest cluster before each merge step. Formally, let $d = (d_1, ..., d_k)$ be the monotonically increasing sequence of inter-cluster distances at which a merge occured. The first discrete derivative d' of this sequence gives the step sizes while the second derivative d'' describes the change rate in step sizes. The index of the maximum in d'' yiels the number of clusters. Note that we require $k \geq 8$ for determining at least two points in the second discrete derivative. In case of $k < 8$, we fall back to the first derivative (step size). Examples show that this is a good approximation for a low number of points. As special cases, since the decision to duplicate is assumed to be already given by the classifier, we exclude the possibility of returning only a single cluster. if $\|\mathcal{N}(v)\| = 2$, then we always trivially split.

[§] Indeed, Ostaszewski et al. applied GO semantic similarity to cluster nodes in a disease map.

[¶] This is basically the *elbow method* in which we plot clustering quality against number of clusters and aim to find an elbow (i.e. bend) in the curve. Clustering quality here would be the minimum inter-cluster distance of two yet unmerged clusters.

Note that thus we characterise the procedure to identify the number of clusters independently of the concrete scale of the data. However, this procedure struggles if intra-cluster variances are diverse an the internal variance of a cluster is close to another inter-cluster distance. We accept this disadvantage for now and hypothesize it has little practical impact in this use-case.

5 Experiments & Results

5.1 Basic hyperparameter search

The models were trained on the AlzPathway reorganisation steps (AʟᴢPᴀᴛʜᴡᴀʏRᴇᴏʀɢ) and evaluated on the Parkinson's Disease Map (PDMᴀᴘ).

Support Vector Machine

For the choice of kernel function, we pick the RBF kernel (see **??**) as a heuristic choice since Nielsen et al. showed that a generally best-performing kernel cannot easily be determined and showed that the RBF kernel generally achieves good performance. Since we use a different SVM implementation than Nielsen et al., that might potentially interpret the parameters slightly differently, for the choice of C and γ, we additionally perform grid search over the same ranges of values. The results are given in Table **??**.

Table 5.1: Considered hyperparameters, value range searched via grid search and best identified value.

Hyperparameter	Searched range	Choice
Cost (C)	2^{-9} to 2^3	0.5
Gamma (γ)	2^{-3} to 2^9	0.1

Graph Neural Network

The range of searched hyperparameters is based on the work of You, Ying, and Leskovec who evaluated choices of hyperparameters for various tasks [81]. They suggest a constrained design space for Graph Neural Networks for tasks such as node classification.

Table 5.2: Considered hyperparameters for the GNN models.

Hyperparameter	Searched range	Choice
Activation function σ	`[PReLU]`	
Use batch normalisation?	`[yes]`	
Dropout	`[0.0, 0.1,0.2,0.4]`	`0.0`
Aggregation function	`[add, mean, max]`	`add`
Fully-connected layers before message-passing	`[1,2]`	`2`
Fully-connected layers after message-passing	`[2,3]`	`2`
Message-passing layers	`[2,4,6,8]`	`2`
Connectivity	`[skip_sum, skip_cat]`	`skip_sum`
Learning rate η	`[0.01]`	
Optimiser	`[adam]`	

5.2 Reproducing previous work

5.3 Handling unbalanced classes

There are two simple ways to approach this problem. One is to modify the training data such that it is balanced. This can be done either by *oversampling* the minory class (coming up with new examples for

the minority class) or by *undersampling* the majority class (dropping examples from the majority class) of the majority class from the training data. For oversampling, we can either simply duplicate examples or synthetically create new examples. We avoid simply duplicating minority class examples because with respect to the classifiers used in this work, we deem this approach analogous to supplying balancing class weight coefficients to the classifier. When synthetically generating new examples, we of course have to settle on a procedure to do so and trust in that it will supply realistic training examples. A common technique for tabular data is SMOTE, in which, for a given data point $\mathbf{x}$, one of its k-nearest neighbours $\mathbf{x}'$ is selected and a new data point is generated by interpolating between $\mathbf{x}$ and $vec x'$, i.e. $\mathbf{x}_{\text{new}} = \mathbf{x} + \lambda(\mathbf{x} - \mathbf{x}')$ where λ is a random choice from $[0, 1]$. However, note that this does not suffice for graph structured data: In addition to attributes, we also have to impute network connectivity. While there exist methods to achieve this (e.g. an extension of SMOTE to network structured-data), we did not explore this approach further due to time constraints and more promising alternatives.

Undersampling   ...

Weights   ... If a high weight is specified, misclassifications of positive examples will be penalised more heavily than misclassificaitions of negative examples.

In case of neural networks, we can introduce an additional coefficient to the loss function that will determine the weight of prediction outcome of the positive class.

We consider the *Weighted Binary Cross Entropy* loss as an extension of **??**, given by

$$\mathcal{L}_{\text{BCE weighted}} = 1/n \sum_{i=1}^{n} w_i y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i) \tag{5.1}$$
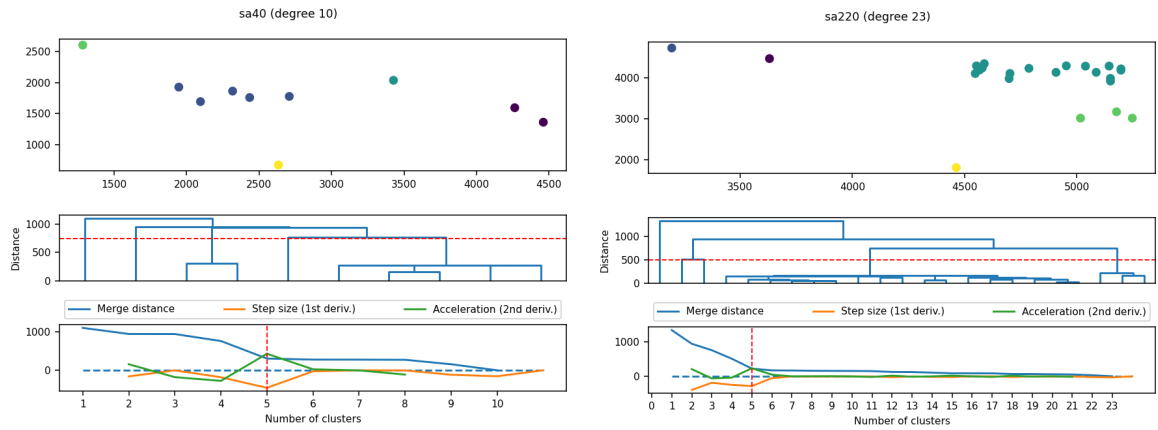
5.4 Graph interpretation for message-passing

5.5 Attention mechanism

5.6 Feature Selection

5.7 Selection of training data

5.8 Attachment of Edges

We provide some examples for successful and problematic cases in Figure **??**. From manual inspection, the vast majority of results seemed to be reasonable clusterings, and in rare cases, ambiguous situations like illustrated here occured.

(a) Example for which the heuristic yields an intuitive clustering. Note that outliers do not distort the clustering result and are assigned their own cluster.

(b) Example in which using the second derivative yields a clustering that intuitively seems more reasonable. The first derivative has its minimum at 2 clusters.
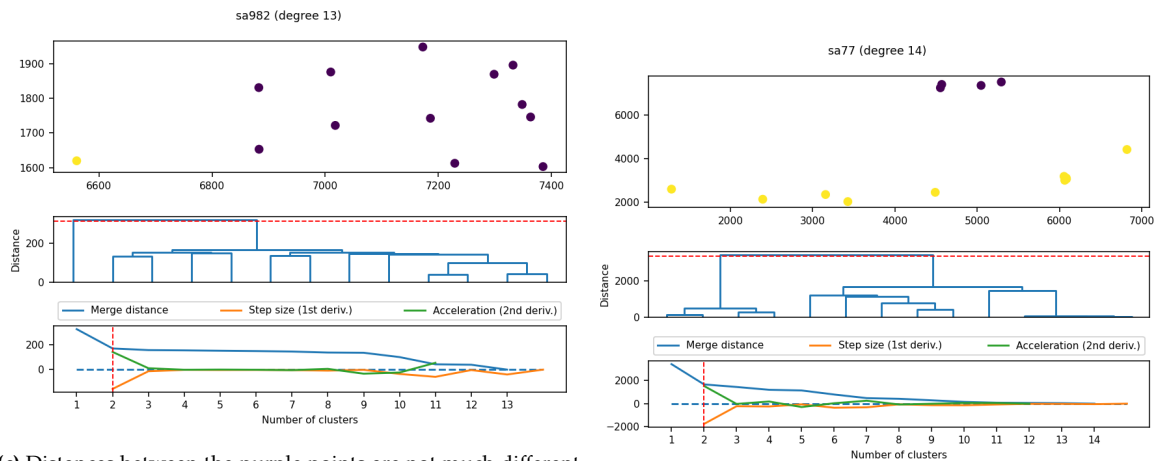
(c) Distances between the purple points are not much different from the distance between the yellow point and the purple cluster (measured by *single linkage*). While the heuristic yields only two clusters, it may have also been feasible to create many clusters. Indeed, the second derivative curve shows another local maximum at 11 clusters.

(d) Another example where large intra-cluster variance is problematic. Contrary to the heuristic's output, it may be preferrable to further split the yellow cluster.

Figure 5.1: Examples for the clustering procedure described in Section **??**. The topmost scatterplot shows the positions of neighbours of a given node in the layout (given node and edges are not shown). Below, a clustering dendrogram describes the distances between two any clusters as they were merged. Finally, the sequence of merge distances as well as its first and second derivative. Taking the minumum, resp. maximum yields a hint for the number of clusters to choose. The red line indicates the suggested choice for the number of clusters and thus a concrete clustering, determing the merge node in the dendrogram where the "cut" should be made. All examples are from AlzPathwayReorg .

6 Discussion

7 Outlook & Future Work

APPENDIX

.1 Notation & Abbreviations

► Vectors and matrices are usually set in bold letters, e.g. $\mathbf{x}$ or $\mathbf{A}$. Vectors are assumed to column vectors unless otherwise specified. $\cdot^T$ denotes the transpose.

► G commonly denotes a graph, $V(G)$ is its vertex set and $E(G)$ its edge set. Whether we consider a directed, undirected, bipartite or simple graph will be made clear from context.

► $\mathcal{N}(v)$ denotes the graph neighbourhood of node v. Unless otherwise specified, this is the direct 1-hop neighbourhood.

► The *bipartite projection* of a bipartite graph with node set $V = A \uplus B$ that is the disjoint union of species aliases A and reactions B, the *bipartite projection onto A* is the simple graph with node set A in which $v_i, v_j \in A$ are connected if and only if in the bipartite they have a common neighbour in B.

► A *species* is an actor in a biological system. Examples for species are proteins, smaller molecules, but also drugs or phenotypes. A *species alias* is a visual representation of a species in a drawing. There can be multiple species aliases corresponding to the same species. A *complex species alias* represents a collection of species aliases, commonly representing protein complexes.

References

[1] Albert-László Barabási and Zoltán N. Oltvai. 'Network Biology: Understanding the Cell's Functional Organization'. In: *Nature Reviews Genetics* 5.2 (2 Feb. 2004), pp. 101–113. DOI: `10.1038/nrg1272`. (Visited on 10/04/2021) (cited on page 2).

[2] Camilo Ruiz, Marinka Zitnik, and Jure Leskovec. 'Identification of Disease Treatment Mechanisms through the Multiscale Interactome'. In: *Nature Communications* 12.1 (1 Mar. 19, 2021), p. 1796. DOI: `10.1038/s41467-021-21770-8`. (Visited on 03/22/2021) (cited on page 2).

[3] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. 'Network Medicine: A Network-Based Approach to Human Disease'. In: *Nature Reviews Genetics* 12.1 (1 Jan. 2011), pp. 56–68. DOI: `10.1038/nrg2918`. (Visited on 10/04/2021) (cited on page 2).

[4] Marek Ostaszewski et al. 'Community-Driven Roadmap for Integrated Disease Maps'. In: *Briefings in Bioinformatics* 20.2 (Mar. 25, 2019), pp. 659–670. DOI: `10.1093/bib/bby024`. (Visited on 06/08/2021) (cited on page 2).

[5] Alexander Mazein et al. 'Systems Medicine Disease Maps: Community-Driven Comprehensive Representation of Disease Mechanisms'. In: *NPJ systems biology and applications* 4 (2018), p. 21. DOI: `10.1038/s41540-018-0059-y` (cited on page 2).

[6] Donna Maglott et al. 'Entrez Gene: Gene-Centered Information at NCBI'. In: *Nucleic Acids Research* 33 (Database issue Jan. 1, 2005), pp. D54–58. DOI: `10.1093/nar/gki031` (cited on pages 3, 5).

[7] The UniProt Consortium. 'UniProt: The Universal Protein Knowledgebase in 2021'. In: *Nucleic Acids Research* 49.D1 (Jan. 8, 2021), pp. D480–D489. DOI: `10.1093/nar/gkaa1100`. (Visited on 10/04/2021) (cited on pages 3, 5).

[8] Stephen G Kobourov. 'Force-Directed Drawing Algorithms'. In: *Handbook of Graph Drawing and Visualization* (2013), p. 26. DOI: `10.1201/b15385-15` (cited on page 3).

[9] Emden R. Gansner, Yehuda Koren, and Stephen North. 'Graph Drawing by Stress Majorization'. In: *Graph Drawing*. Ed. by János Pach. Red. by David Hutchison et al. Vol. 3383. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 239–250. DOI: `10.1007/978-3-540-31843-9_25`. (Visited on 03/31/2020) (cited on pages 3, 14).

[10] Patrick Healy and Nikola S Nikolov. 'Hierarchical Drawing Algorithms'. In: *Handbook of Graph Drawing and Visualization* (2013), p. 46 (cited on page 3).

[11] Christian A Duncan and Michael T Goodrich. 'Planar Orthogonal and Polyline Drawing Algorithms'. In: *Handbook of Graph Drawing and Visualization* (2013), p. 24 (cited on page 3).

[12] Martin Siebenhaller et al. 'Human-like Layout Algorithms for Signalling Hypergraphs: Outlining Requirements'. In: *Briefings in Bioinformatics* 21.1 (Jan. 17, 2020), pp. 62–72. DOI: `10.1093/bib/bby099`. (Visited on 06/08/2021) (cited on page 3).

[13] Romain Bourqui et al. 'Metabolic Network Visualization Eliminating Node Redundance and Preserving Metabolic Pathways'. In: *BMC Systems Biology* 1.1 (July 3, 2007), p. 29. DOI: `10.1186/1752-0509-1-29`. (Visited on 09/23/2021) (cited on page 3).

[14] Steve Kieffer et al. 'HOLA: Human-like Orthogonal Network Layout'. In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (Jan. 31, 2016), pp. 349–358. DOI: `10.1109/TVCG.2015.2467451`. (Visited on 06/08/2021) (cited on page 3).

[15] Hsiang-Yun Wu et al. 'Metabopolis: Scalable Network Layout for Biological Pathway Diagrams in Urban Map Style'. In: *BMC Bioinformatics* 20.1 (Apr. 15, 2019), p. 187. DOI: `10.1186/s12859-019-2779-4`. (Visited on 09/23/2021) (cited on page 3).

[16] Mikael Huss and Petter Holme. 'Currency and Commodity Metabolites: Their Identification and Relation to the Modularity of Metabolic Networks'. In: *IET Systems Biology* 1.5 (Sept. 1, 2007), pp. 280–285. DOI: `10.1049/iet-syb:20060077`. (Visited on 06/09/2021) (cited on pages 4, 13).

[17] Eun-Youn Kim, Daniel Ashlock, and Sung Ho Yoon. 'Identification of Critical Connectors in the Directed Reaction-Centric Graphs of Microbial Metabolic Networks'. In: *BMC bioinformatics* 20.1 (June 13, 2019), p. 328. DOI: 10.1186/s12859-019-2897-z (cited on page 4).

[18] David S. Wishart et al. 'DrugBank: A Knowledgebase for Drugs, Drug Actions and Drug Targets'. In: *Nucleic Acids Research* 36 (suppl_1 Jan. 1, 2008), pp. D901–D906. DOI: 10.1093/nar/gkm958. (Visited on 10/05/2021) (cited on page 5).

[19] Michael Ashburner et al. 'Gene Ontology: Tool for the Unification of Biology'. In: *Nature Genetics* 25.1 (1 May 2000), pp. 25–29. DOI: 10.1038/75556. (Visited on 10/05/2021) (cited on page 5).

[20] Rex Ying et al. *Hierarchical Graph Representation Learning with Differentiable Pooling*. Feb. 20, 2019. URL: http://arxiv.org/abs/1806.08804 (visited on 02/04/2021) (cited on pages 5, 7).

[21] Aston Zhang et al. 'Dive into Deep Learning'. In: (), p. 1021 (cited on pages 5, 6, 8).

[22] Vladimir Vapnik. 'Principles of Risk Minimization for Learning Theory'. In: (), p. 8 (cited on page 5).

[23] Michael M. Bronstein et al. *Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges*. May 2, 2021. URL: http://arxiv.org/abs/2104.13478 (visited on 05/06/2021) (cited on page 5).

[24] Lex Flagel, Yaniv Brandvain, and Daniel R. Schrider. 'The Unreasonable Effectiveness of Convolutional Neural Networks in Population Genetic Inference'. In: *Molecular Biology and Evolution* 36.2 (Feb. 1, 2019), pp. 220–238. DOI: 10.1093/molbev/msy224 (cited on page 6).

[25] Genta Aoki and Yasubumi Sakakibara. 'Convolutional Neural Networks for Classification of Alignments of Non-Coding RNA Sequences'. In: *Bioinformatics* 34.13 (July 1, 2018), pp. i237–i244. DOI: 10.1093/bioinformatics/bty228. (Visited on 10/06/2021) (cited on page 6).

[26] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. Sept. 10, 2018. URL: http://arxiv.org/abs/1706.02216 (visited on 06/16/2021) (cited on page 7).

[27] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. Feb. 22, 2017. URL: http://arxiv.org/abs/1609.02907 (visited on 12/07/2020) (cited on page 7).

[28] Petar Veličković et al. *Graph Attention Networks*. Feb. 4, 2018. URL: http://arxiv.org/abs/1710.10903 (visited on 01/05/2021) (cited on page 8).

[29] Ashish Vaswani et al. *Attention Is All You Need*. Dec. 5, 2017. URL: http://arxiv.org/abs/1706.03762 (visited on 10/10/2021) (cited on page 8).

[30] Taocheng Liu, Xiangbin Ye, and Bei Sun. 'Combining Convolutional Neural Network and Support Vector Machine for Gait-Based Gender Recognition'. In: *2018 Chinese Automation Congress (CAC)*. 2018 Chinese Automation Congress (CAC). Nov. 2018, pp. 3477–3481. DOI: 10.1109/CAC.2018.8623118 (cited on page 8).

[31] Sami Tibshirani, Harry Friedman, and Trevor Hastie. 'The Elements of Statistical Learning'. In: (2017), p. 764 (cited on page 9).

[32] Sune S. Nielsen et al. 'Machine Learning to Support the Presentation of Complex Pathway Graphs'. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2019), pp. 1–1. DOI: 10.1109/TCBB.2019.2938501 (cited on pages 12, 13, 16–19, 24).

[33] Akira Funahashi et al. 'CellDesigner 3.5: A Versatile Modeling Tool for Biochemical Networks'. In: *Proceedings of the IEEE* 96.8 (Aug. 2008), pp. 1254–1265. DOI: 10.1109/JPROC.2008.925458 (cited on page 12).

[34] Piotr Gawron et al. 'MINERVA—a Platform for Visualization and Curation of Molecular Interaction Networks'. In: *npj Systems Biology and Applications* 2.1 (1 Sept. 22, 2016), pp. 1–6. DOI: 10.1038/npjsba.2016.20. (Visited on 10/04/2021) (cited on page 12).

[35] Paul Shannon et al. 'Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks'. In: *Genome Research* 13.11 (Jan. 11, 2003), pp. 2498–2504. DOI: 10.1101/gr.1239303. (Visited on 05/04/2021) (cited on page 12).

[36] Hendrik Rohn et al. 'VANTED v2: A Framework for Systems Biology Applications'. In: *BMC Systems Biology* 6.1 (2012), p. 139. DOI: 10.1186/1752-0509-6-139. (Visited on 04/24/2019) (cited on page 12).

[37]   Annegret Liebers. 'Planarizing Graphs — A Survey and Annotated Bibliography'. In: (2001), p. 74 (cited on page 12).

[38]   Faisal N. Abu-Khzam et al. 'Cluster Editing with Vertex Splitting'. In: *Combinatorial Optimization*. Ed. by Jon Lee, Giovanni Rinaldi, and A. Ridha Mahjoub. Vol. 10856. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 1–13. DOI: `10.1007/978-3-319-96151-4_1`. (Visited on 10/13/2021) (cited on page 12).

[39]   P. Eades and C. F. X. de Mendonça N. 'Vertex Splitting and Tension-Free Layout'. In: *Graph Drawing*. Ed. by Franz J. Brandenburg. Red. by G. Goos, J. Hartmanis, and J. van Leeuwen. Vol. 1027. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 202–211. DOI: `10.1007/BFb0021804`. (Visited on 06/16/2021) (cited on page 12).

[40]   Tomihisa Kamada and Satoru Kawai. 'An Algorithm for Drawing General Undirected Graphs'. In: *Information Processing Letters* 31.1 (1989), p. 9 (cited on page 12).

[41]   J. Li. *Eliminate Wire Crossings by Node Duplication*. 2008. URL: `https://www.semanticscholar.org/paper/Eliminate-Wire-Crossings-by-Node-Duplication-Li/f15e6d3a207ca83339ee1901047a6d7905cc4d39` (visited on 09/23/2021) (cited on page 12).

[42]   Doo-Kwon Paik, S. Reddy, and S. Sahni. 'Vertex Splitting in Dags and Applications to Partial Scan Designs and Lossy Circuits'. In: *Int. J. Found. Comput. Sci.* (1998). DOI: `10.1142/S0129054198000301` (cited on page 12).

[43]   M. Mayer and F. Erçal. 'Genetic Algorithms for Vertex Splitting in DAGs'. In: *undefined* (1993). (Visited on 10/13/2021) (cited on page 12).

[44]   N. Henr, A. Bezerianos, and J.-D. Fekete. 'Improving the Readability of Clustered Social Networks Using Node Duplication'. In: *IEEE Transactions on Visualization and Computer Graphics* 14.6 (Nov. 2008), pp. 1317–1324. DOI: `10.1109/TVCG.2008.141`. (Visited on 06/08/2021) (cited on page 12).

[45]   Ichcha Manipur et al. 'Clustering Analysis of Tumor Metabolic Networks'. In: *BMC Bioinformatics* 21.10 (Aug. 25, 2020), p. 349. DOI: `10.1186/s12859-020-03564-9`. (Visited on 01/06/2021) (cited on pages 12, 13).

[46]   H. Ma and A. Zeng. 'Reconstruction of Metabolic Networks from Genome Data and Analysis of Their Global Structure for Various Organisms'. In: *undefined* (2003). (Visited on 05/28/2021) (cited on page 13).

[47]   S. Schuster et al. 'Exploring the Pathway Structure of Metabolism: Decomposition into Subnetworks and Application to Mycoplasma Pneumoniae'. In: *Bioinformatics* 18.2 (Feb. 1, 2002), pp. 351–361. DOI: `10.1093/bioinformatics/18.2.351`. (Visited on 11/20/2020) (cited on page 13).

[48]   M. E. J. Newman. 'Modularity and Community Structure in Networks'. In: *Proceedings of the National Academy of Sciences* 103.23 (June 6, 2006), pp. 8577–8582. DOI: `10.1073/pnas.0601602103`. (Visited on 04/12/2019) (cited on page 13).

[49]   Roger Guimerà and Luís A. Nunes Amaral. 'Functional Cartography of Complex Metabolic Networks'. In: *Nature* 433.7028 (7028 Feb. 2005), pp. 895–900. DOI: `10.1038/nature03288`. (Visited on 06/09/2021) (cited on page 13).

[50]   Markus Rohrschneider et al. 'A Novel Grid-Based Visualization Approach for Metabolic Networks with Advanced Focus&Context View'. In: *Graph Drawing*. Ed. by David Eppstein and Emden R. Gansner. Red. by David Hutchison et al. Vol. 5849. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 268–279. DOI: `10.1007/978-3-642-11805-0_26`. (Visited on 06/11/2021) (cited on page 13).

[51]   G. Joshi-Tope et al. 'Reactome: A Knowledgebase of Biological Pathways'. In: *Nucleic Acids Research* 33 (suppl_1 Jan. 1, 2005), pp. D428–D432. DOI: `10.1093/nar/gki072`. (Visited on 10/13/2021) (cited on page 13).

[52]   Alice C. Villéger, Stephen R. Pettifer, and Douglas B. Kell. 'Arcadia: A Visualization Tool for Metabolic Pathways'. In: *Bioinformatics* 26.11 (June 1, 2010), pp. 1470–1471. DOI: `10.1093/bioinformatics/btq154`. (Visited on 10/04/2021) (cited on page 13).

[53] Peter Droste, Katharina Nöh, and Wolfgang Wiechert. 'Omix - A Visualization Tool for Metabolic Networks with Highest Usability and Customizability in Focus'. In: *Chemie Ingenieur Technik* 85.6 (June 2013), pp. 849–862. DOI: 10.1002/cite.201200234. (Visited on 10/04/2021) (cited on page 13).

[54] Soichi Ogishima et al. 'AlzPathway, an Updated Map of Curated Signaling Pathways: Towards Deciphering Alzheimer's Disease Pathogenesis'. In: *Methods in Molecular Biology (Clifton, N.J.)* 1303 (2016), pp. 423–432. DOI: 10.1007/978-1-4939-2627-5_25 (cited on pages 13, 16).

[55] Kazuhiro A. Fujita et al. 'Integrating Pathways of Parkinson's Disease in a Molecular Interaction Map'. In: *Molecular Neurobiology* 49.1 (Feb. 1, 2014), pp. 88–102. DOI: 10.1007/s12035-013-8489-4. (Visited on 06/14/2021) (cited on pages 13, 16).

[56] Marek Ostaszewski et al. 'COVID-19 Disease Map, Building a Computational Repository of SARS-CoV-2 Virus-Host Interaction Mechanisms'. In: *Scientific Data* 7.1 (May 5, 2020), p. 136. DOI: 10.1038/s41597-020-0477-8 (cited on page 13).

[57] Satoshi Mizuno et al. 'Network Analysis of a Comprehensive Knowledge Repository Reveals a Dual Role for Ceramide in Alzheimer's Disease'. In: *PloS One* 11.2 (2016), e0148431. DOI: 10.1371/journal.pone.0148431 (cited on pages 13, 16).

[58] Nicolas Sompairac et al. 'Metabolic and Signalling Network Maps Integration: Application to Cross-Talk Studies and Omics Data Analysis in Cancer'. In: *BMC Bioinformatics* 20 (Suppl 4 Apr. 18, 2019). DOI: 10.1186/s12859-019-2682-z. (Visited on 04/14/2021) (cited on page 13).

[59] Matteo Tiezzi, Gabriele Ciravegna, and Marco Gori. *Graph Neural Networks for Graph Drawing*. Sept. 21, 2021. URL: http://arxiv.org/abs/2109.10061 (visited on 10/13/2021) (cited on page 14).

[60] Ronghui You et al. 'DeepGraphGO: Graph Neural Network for Large-Scale, Multispecies Protein Function Prediction'. In: *Bioinformatics* 37 (Supplement_1 July 1, 2021), pp. i262–i271. DOI: 10.1093/bioinformatics/btab270. (Visited on 10/13/2021) (cited on page 14).

[61] Xiaoshi Zhong, Rama Kaalia, and Jagath C. Rajapakse. 'GO2Vec: Transforming GO Terms and Proteins to Vector Representations via Graph Embeddings'. In: *BMC Genomics* 20.9 (Feb. 18, 2020), p. 918. DOI: 10.1186/s12864-019-6272-2. (Visited on 09/06/2021) (cited on pages 14, 20).

[62] Neal Ravindra et al. 'Disease State Prediction from Single-Cell Data Using Graph Attention Networks'. In: *Proceedings of the ACM Conference on Health, Inference, and Learning*. CHIL '20. New York, NY, USA: Association for Computing Machinery, Apr. 2, 2020, pp. 121–130. DOI: 10.1145/3368555.3384449. (Visited on 04/21/2021) (cited on page 14).

[63] Jonathan M. Stokes et al. 'A Deep Learning Approach to Antibiotic Discovery'. In: *Cell* 180.4 (Feb. 2020), 688–702.e13. DOI: 10.1016/j.cell.2020.01.021. (Visited on 10/14/2021) (cited on pages 14, 15).

[64] David Duvenaud et al. *Convolutional Networks on Graphs for Learning Molecular Fingerprints*. Nov. 3, 2015. URL: http://arxiv.org/abs/1509.09292 (visited on 01/21/2021) (cited on page 15).

[65] Oliver Wieder et al. 'A Compact Review of Molecular Property Prediction with Graph Neural Networks'. In: *Drug Discovery Today: Technologies* (Dec. 17, 2020). DOI: 10.1016/j.ddtec.2020.11.009. (Visited on 10/14/2021) (cited on page 15).

[66] Thomas Gaudelet et al. *Utilising Graph Machine Learning within Drug Discovery and Development*. Dec. 9, 2020. URL: http://arxiv.org/abs/2012.05716 (visited on 01/14/2021) (cited on page 15).

[67] Mayank Baranwal et al. 'A Deep Learning Architecture for Metabolic Pathway Prediction'. In: *Bioinformatics (Oxford, England)* 36.8 (Apr. 15, 2020), pp. 2547–2553. DOI: 10.1093/bioinformatics/btz954 (cited on page 15).

[68] Bajaj, Heereguppe, and Sumath. *Graph Convolutional Networks to Explore Drug and Disease Relationships in Biological Networks*. 2017. (Visited on 10/14/2021) (cited on page 15).

[69] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 'Modeling Polypharmacy Side Effects with Graph Convolutional Networks'. In: *Bioinformatics* 34.13 (July 1, 2018), pp. i457–i466. DOI: 10.1093/bioinformatics/bty294. (Visited on 12/14/2020) (cited on page 15).

[70] Hryhorii Chereda et al. 'Explaining Decisions of Graph Convolutional Neural Networks: Patient-Specific Molecular Subnetworks Responsible for Metastasis Prediction in Breast Cancer'. In: *Genome Medicine* 13 (Mar. 11, 2021), p. 42. DOI: 10.1186/s13073-021-00845-7. (Visited on 10/14/2021) (cited on page 15).

[71] Xiao-Meng Zhang et al. 'Graph Neural Networks and Their Current Applications in Bioinformatics'. In: *Frontiers in Genetics* 12 (July 29, 2021), p. 690049. DOI: 10.3389/fgene.2021.690049. (Visited on 10/14/2021) (cited on page 15).

[72] Vincent Henry et al. 'Converting Disease Maps into Heavyweight Ontologies: General Methodology and Application to Alzheimer's Disease'. In: *Database: The Journal of Biological Databases and Curation* 2021 (Feb. 16, 2021), baab004. DOI: 10.1093/database/baab004 (cited on page 15).

[73] Marek Ostaszewski et al. 'Clustering Approaches for Visual Knowledge Exploration in Molecular Interaction Networks'. In: *BMC Bioinformatics* 19.1 (Aug. 29, 2018), p. 308. DOI: 10.1186/s12859-018-2314-z. (Visited on 06/07/2021) (cited on pages 15, 22).

[74] Satoshi Mizuno et al. 'AlzPathway: A Comprehensive Map of Signaling Pathways of Alzheimer's Disease'. In: *BMC Systems Biology* 6.1 (May 30, 2012), p. 52. DOI: 10.1186/1752-0509-6-52. (Visited on 06/02/2021) (cited on page 16).

[75] S. Ogishima et al. 'A Map of Alzheimer's Disease-Signaling Pathways: A Hope for Drug Target Discovery'. In: *Clinical Pharmacology and Therapeutics* 93.5 (May 2013), pp. 399–401. DOI: 10.1038/clpt.2013.37 (cited on page 16).

[76] Marek Ostaszewski. *AlzPathway Regorganisation Steps for Increased Modularity*. Zenodo, June 25, 2021. DOI: 10.5281/zenodo.5032278. (Visited on 06/28/2021) (cited on page 16).

[77] Alberto Noronha et al. 'ReconMap: An Interactive Visualization of Human Metabolism'. In: *Bioinformatics* 33.4 (Feb. 15, 2017), pp. 605–607. DOI: 10.1093/bioinformatics/btw667. (Visited on 07/09/2021) (cited on page 16).

[78] Ines Thiele et al. 'A Community-Driven Global Reconstruction of Human Metabolism'. In: *Nature Biotechnology* 31.5 (5 May 2013), pp. 419–425. DOI: 10.1038/nbt.2488. (Visited on 10/18/2021) (cited on page 16).

[79] *CellDesigner 4 Extension Tag Specification Document*. 2010. URL: http://www.celldesigner.org/documents/CellDesigner4ExtensionTagSpecificationE.pdf (visited on 08/16/2021) (cited on page 17).

[80] Ulrik Brandes and Thomas Erlebach, eds. *Network Analysis: Methodological Foundations*. LCNS, Tutorial 3418. Berlin ; New York: Springer, 2005. 471 pp. (cited on page 19).

[81] Jiaxuan You, Rex Ying, and Jure Leskovec. *Design Space for Graph Neural Networks*. Nov. 17, 2020. URL: http://arxiv.org/abs/2011.08843 (visited on 02/26/2021) (cited on page 24).