Chapter 2: Background

The current work massively builds on the notion of *data clustering*. This fundamental task has been a subject for extensive study of computational disciplines such as machine learning and pattern recognition and is useful for a variety of applications in many empirical domains. The review below intends to provide a broad perspective on the methods that are elaborated later on. The original parts of our work introduced in later chapters elaborate on two particular data clustering methods, which are described in more detail therein. Chapter 4 below is based on Puzicha, Hofmann & Buhmann's (2000) *theory of proximity based clustering*. Chapter 5 is based on two closely related data clustering methods: the *information bottleneck* method (IB; Tishby, Pereira & Bialek, 1999) and the *information distortion* method (ID, Gedeon, Parker & Dimitrov, 2003; Pereira, Tishby & Lee, 1993).

This work is largely inspired as well, though in a less technical level, by computational models of analogy – a field of study in cognitive science, on the borderline between linguistics, psychology and the computational sciences. This chapter therefore concludes with an exemplifying discussion of this field.

2.1 Data Clustering

Data clustering is a computational task, which aims at revealing structure in initially unstructured data. The following definition by Aldenderfer & Blashfield (1984),

The segmentation of heterogeneous set of elements into a collection of homogenous subsets,

provides a good notion of what data clustering is about.

2.1.1 Introduction

The ability to cluster data might be useful for the functioning of intelligent systems, whether artificial or natural. Two closely related motivations for why intelligent systems are expected to develop data-clustering capabilities are (after MacKay, 2003, Ch. 20):

- **Prediction**: characterizing newly encountered pieces of information as exemplars of identifiable distinct classes.
- **Communication**: by referring to similar pieces of information by a common cluster label, communicating parties can concentrate on what is important to the communicated information, while avoiding unneeded details and subtleties.

The above two considerations are in fact closely related. For example, we might expect an intelligent system encountering an object to be able to identify it as, say, 'an animal', 'a plant', or 'a piece of furniture' and, at a finer level of resolution, as 'a cat' or 'a tiger'. Such labeling system might play a role in classifying data both for the system's own purposes and for communicating this information to other parties.

Data clustering methods approach the above restrictedly from unsupervised perspective. Likewise, the computational mechanisms reviewed and developed throughout this work rely on unlabeled training data, while external supervision providing known-to-be "correct labels" is not introduced during learning (as is indeed the case in many practical settings).

2.1.1.1 Practical Applications

There are many applicative uses for data clustering. A partial illustrative list is as follows:

- **Object recognition**: partitioning of a set of images, recorded sounds, or other composite records that are not subject to immediate automated interpretation, into well-distinguished classes. The intended scope of each "well-distinguished class" may vary. Few examples are:
  - all exemplars of images from the same natural category (e.g., "animals" in one cluster, "buildings" in another one and "pieces of furniture" in a third cluster).
  - all exemplars of the same object photographed under different conditions (changing illumination, angle etc.)
  - all exemplars of recorded pronunciations of the same word (e.g., names of alphabet letters, as in Blatt, Wiseman, & Domany, 1997)
- **Image segmentation**: the classifications of image pixels to different textures or to different objects (Hofmann & Puzicha, 1998; Boykov, Vexler & Zabih, 1999), e.g. objects in a room or organs in an image produced by some medical imaging technique.
- **Information Retrieval and Natural Language Processing**: topical categorization of documents (Dhillon & Modha, 2001; Slonim & Tishby, 2002); detecting senses and sub-topics through word clusters (Pereira, Tishby & Lee, 1993; Brew & Schulte im Walde, 2002; Korhonen, Krymolowski & Marx, 2003).

Forming clusters of semantically similar documents or words is in particular related with the applicative part of the current work. Word clusters illustrate how what can be called "a conceptual network", where each cluster of words sharing a common sense reveals a different concept, emerges from unprocessed textual data. This is accomplished with no supervision, not to mention "deep understanding" of the language the analyzed texts are written in. In our case, as well as in other

works such as the two above references on document clustering, clusters are produced with no information about the texts except word co-occurrence statistics (the three works on word clustering cited above use syntactic information extracted automatically prior to the clustering). Thus, beyond its applicative utility data clustering can be seen as demonstrating, on a very basic level, computation-based intelligence and the emergence of meaning.

2.1.1.2 Data Clustering is an Ill-posed Task

A well-known fact about data clustering is that it is *ill posed*, or, as Estivill-Castro (2003) puts it: "the cluster is in the eyes of the beholder". Any definition to the data-clustering problem does not set an unambiguous criterion to judging whether, or how well, the problem is solved. Quite often, several partitions exist representing compromises between different biases inherent to the data, so the user must be more detailed with regard to the specific goals and the considerations relevant to those goals. For instance, image segmentation data is constrained not only by pixels' color or grey scale, but also by the spatial proximity of the clustered pixels. Spatial proximity, however, is irrelevant in most other settings. Being more specific and posing more restrictions does not necessarily turns the data clustering into a well-defined problem. Assuming an axiomatic framework that poses some more concrete restrictions, Kleinberg (2003; see Subsection 2.1.4.3 below) shows that the data clustering problem is inherently unsatisfiable.

2.1.2 The Structure of Data Clustering Output

Basically, a data clustering method is supposed, given a set of elements, to produce a partition of the set into clusters. We use the term *clustering configuration* to denote any one of all possible partitions of the data, among which data clustering methods are supposed to identify the one that optimally addresses the data-clustering task. Each cluster is, by definition, just the set of data elements that forms it. There are, however, methods that provide also supplementary information. There are several methods that specify also a *prototypical representation* for every cluster, which gives an idea about each cluster's characteristics while saving the need to examine its elements. We elaborate on such cases of extended clustering output later on (see 2.1.4.4). In this subsection, we discuss two issues that are part of the clustering output in its more basic form: the number of clusters being produced and multi assignments and probabilistic assignments of data elements to clusters.

2.1.2.1 How Many Clusters

It is quite clear that the number of clusters, k, is expected to be considerably less than the number of clustered elements. However, determining k exactly is a non-trivial issue, which might depend, for instance, on specific user requirements. There are methods (e.g., Blatt, Wiseman & Domany, 1997) that infer k from the data itself, with no directions regarding user preferences. There are also

statistical significance tests comparing configurations of k versus k+1 clusters in order to judge whether producing the larger number of clusters is justified (Duda, Hart & Strock, 2001, pp. 557-559). Many methods, however, including those adapted in the current work, follow user instructions. In these cases, k is usually specified as an additional input parameter. Alternatively, some methods require the user to indicate the value of a method-specific threshold parameter, which indirectly determines k, with dependence on the data.

2.1.2.2 Assignment Probabilities

In many situations, partitioning the data to homogeneous subsets leaves some elements that do not fit perfectly into one of the clusters. One conceivable solution, sometimes termed "soft" or non-deterministic clustering, is to allow the assignment of an element to two or more clusters, with varying degrees of justification or confidence. Hence, the clustering output of several methods includes detailed information with regard to "level of assignment" $ass(c,x)$ of each element x within each cluster c. A common convention is to restrict the assignment levels to be non-negative, $ass(c,x) \geq 0$, and to require all assignment levels of an individual element x to sum up to 1: $\sum_c ass(c,x) = 1$. Under this convention, the level by which an element is assigned to a cluster can be interpreted as $ass(c,x) \equiv p(c|x)$, the *probability* of the (deterministic) assignment to c to take place, given that the assigned element is x. We call methods with this kind of output *probabilistic clustering* methods. In this work, we use a probabilistic framework in Chapter 5.

2.1.3 Data Representation

Different data clustering methods differ by the type of data representation they are capable of dealing with – the format of the input they can take.

2.1.3.1 Pairwise Representation

Pairwise clustering methods rely on measures of proximity, i.e. similarity or dissimilarity values, between pairs of data elements. An example for a direct source for similarity values utilizable for pairwise clustering is a confusion matrix (e.g., Manos, 1996). This is a matrix, based on the performance of subjects under study, in which both rows and columns correspond to data elements. The entry at a row corresponding to an element x and column corresponding to an element x' indicates empirical count $count(x \leftarrow x')$, or estimated probability $p(x|x')$, of miss-recognizing an element x' as an element x. More on proximity measures in 2.1.3.3 below.

2.1.3.2 Feature-based Representation

Somewhat more typical to actual data than pairwise representation is a representation where, along with the set of elements to be clustered, an auxiliary set of features that are considered relevant to the desired outcome (*relevance features*) is present. In such case, each element is identifiable as a vector

with entries that reflect how intensive the element's association with each of the various features is. For example, an element x can be represented by the vector of observed co-occurrence counts of x with each one of the features y. Such co-occurrence vectors are widely used, including in this work.

There are clustering methods that process co-occurrence vectors in their normalized form, i.e., vectors of estimated conditional probabilities $p(y|x)$ of each feature y to co-occur with an element x. The normalization factor is the total occurrence count of the element x in the data, over all features with which x co-occurs (*count*(x)). In a probabilistic setting, the total counts of data elements are often normalized themselves to a probabilistic vector of relative frequencies $p(x)$, for each data element x.

2.1.3.3 Proximity Measures

Similarity and dissimilarity assessment are practiced both as an independent unsupervised task and as pre-processing for other tasks, including data clustering. It is used within many applications: data mining (Das, Mannila & Ronkainen, 1998), image retrieval (Ortega et al., 1998) and document clustering (Dhillon & Modha, 2001).

Dissimilarity can be viewed as distance in the feature space. The well-known L_1 (*Manhattan*) norm, which is the sum of absolute differences between corresponding vector coordinates

$$L_1(x, x') = \sum_i |x_i - x'_i| , \qquad (2.1)$$

where corresponding coordinates are indexed by a common index i, and the L_2 (*Euclidean*) *distance*, which is the square root of sum of squared coordinate differences

$$L_2(x, x') = \sqrt{\sum_i (x_i - x'_i)^2} , \qquad (2.2)$$

are sometimes used as benchmarks (e.g. by Lee, 1999). In general, there are no strict formal restrictions on the similarity or distance values. For instance, a dissimilarity measure is not expected to form a *metric* as L_1 and L_2 do, and in particular, a distance measure d is not required to be symmetric ($d(x,x') \neq d(x',x)$ is permitted) or to obey the triangle inequality ($d(x,x') + d(x',x'') \leq d(x,x'')$ is permitted).

Several dissimilarity measures refer to representation of data in the above mentioned form of conditional probability distributions of features. The *Kullback-Leibler* (*KL*) *divergence* (Cover & Thomas, 1991, p. 18) quantifies the inefficiency of coding information distributed according to $p(y|x')$ with code that is optimized for $p(y|x)$:

$$KL[p(y|x)||p(y|x')] = \sum_y p(y|x) \, (\log p(y|x) - \log p(y|x')). \qquad (2.3)$$

KL divergence is undefined, i.e. approaches infinity, whenever there is a feature y^* such that $p(y^*|x) > 0$ but $p(y^*|x') = 0$. There are dissimilarity measures that are based on *KL* divergence but overcome this problem. The *Jensen-Shannon* (*JS*) *divergence* (used, e.g., in Manos, 1996, and Korhonen, Krymolowski & Marx, 2003) is the sum of *KL* divergences of $p(y|x)$ and $p(y|x')$ from their average, possibly weighted by the elements' relative frequencies $p(x)$ and $p(x')$:

$$JS[p(y|x)||p(y|x')] = \pi KL[p(y|x)||q(y)] + \pi' KL[p(y|x')||q(y)], \qquad (2.4)$$

where $\pi = p(x) / (p(x) + p(x'))$, $\pi' = p(x') / (p(x) + p(x'))$ and $q(y) = (\pi p(y|x) + \pi' p(y|x'))$. The *ff-skew divergence* (Lee, 1999) is the *KL* divergence between $p(y|x)$ and a slight shift, by a small positive value α, of $p(y|x')$ towards $p(y|x)$:

$$ff\text{-}skew[p(y|x)||p(y|x')] = KL[p(y|x) \,||\, (1-\alpha)p(y|x') + \alpha p(y|x)]. \qquad (2.5)$$

Both the symmetric *JS* divergence and the non-symmetric *ff*-skew divergence are guaranteed not to approach infinity, because the average of $p(y|x)$ and $p(y|x')$, as well as the shifted probability p', are equal to zero only on those features y for which both $p(y|x)$ and $p(y|x')$ are equal to zero.

Similarity values are often induced from distance values. A popular scheme of calculating similarity between two data elements x and x' is through an exponentially decreasing function of their given distance d: $sim(x,x') = e^{-d(x,x')}$ (as, e.g., done by Blatt, Wiseman & Domany, 1997). This scheme tends to sharpen differences between pairs of close neighbors, while blurring differences between pairs of distant elements, thus it intensifies sensitivity to neighborhood-related information that is more relevant to the clustering task (distant elements would always be assigned to different clusters).

In the domain of text processing, co-occurrence based similarity measures are widely used (see Dagan, 2000 for a review). Such measures rely on the observation that semantically similar words tend to share similar patterns of co-occurrences with their neighboring words, which take the role of relevance features in this case. Likewise, similar documents share similar word frequency distributions. A common convention bounds the similarity values between 0 to 1, where 1 typically denotes self-similarity. One widely used measure is the cosine of the angel between two co-occurrence count vectors (van Rijsbergen, 1979, ch. 5; used, e.g., by Dhillon & Modha, 2001), that is, the sum of corresponding entry products (dot product) of the two vectors normalized by the L_2 norm of the product:

$$cosine(x, x') = \frac{\sum_y count(x, y) count(x', y)}{\sqrt{\sum_y count(x, y)^2} \sqrt{\sum_y count(x', y)^2}}, \qquad (2.6)$$

The cosine and other similarly straightforward measures are affected by the data sparseness problem, intensified, for instance, by the common use of different words referring to similar meanings. More sophisticated measures, which are designed to overcome the sparseness problem, incorporate

information-theoretic perspective. We employ in this work two such similarity measures, by Lin (1998) and Dagan, Marcus & Markovitch (1995). These measures are described in detail later, in Chapter 4.

2.1.3.4 Re-representation and Preprocessing

There are cases where the given representation of the data is modified before applying the actual clustering method. This might be required, for instance, because the method in use takes different representation than the given one, or in order to have smaller or less noisy data so that the method operates faster or produces results of better quality.

Re-representation in general implies loss of information. Consequently, an important sub-goal of pre-processing is to preserve the information relevant to the clustering task. When pairwise data is given, similarities between distant elements are often ignored, resulting in sparse similarity matrix, which improves computation efficiency. This is an example of rather straightforward elimination of irrelevant information: as noted, any two unmistakably dissimilar elements are expected to be in different clusters and their exact level of similarity is less important.

A common re-representation procedure is the transformation of feature vectors into an array of similarities, or distances, appropriate for pairwise clustering. Such transformation is based on a measure of distance or similarity between vectors (see previous subsection). Often, terms such as 'similarity' or 'distance' are mentioned also in the context of feature-based clustering, which gets a concrete meaning only if a concrete measure of similarity between feature vectors is specified.

Feature-selection and feature-generation techniques can be applied to convert one form of vectorial representation into another, with aims such as masking noise from the given representation or decreasing the data complexity prior to actual clustering. For example, PCA (*principled component analysis)*, a dimensionality-reduction procedure, is used by Brew & Schulte im Walde (2002) as a pre-processing procedure prior to applying a clustering method.

2.1.4 Algorithmic Framework for Data Clustering

The number of all possible partitions of n elements to k clusters is roughly $k^n/k!$, which grows exponentially with n (Duda, Hart & Stork, 2001, p. 548). Hence, going through all different clustering configurations in search for one realizing the requirements, whatever they are, is computationally ineffective (indeed, data clustering is an NP-complete problem, Garey & Johnson, 1979). Accordingly, clustering algorithms typically attempt to provide, by means of restricted but principled search, a reasonably good solution even if spotting the absolutely best solution in not guaranteed. The following subsections present some schemes for how various data clustering methods conduct such a search.

2.1.4.1 Incremental Search

A strategy for solving optimization problems that are not liable to an exhaustive search, implemented within most data clustering methods that are mentioned below, is performing the task incrementally, as shown in Fig 2.1.

```
given an initial configuration of clusters
repeat
      from a set of currently available update steps of a pre-specified
      type, perform the one that is optimal due to some pre-specified
      criterion, so that a new clustering configuration results
Until the resulting configuration meets some pre-specified stop condition
```

Figure 2.1: The incremental clustering scheme.

There are several simple examples for methods implementing this step-by-step scheme. One well-known class of such methods assumes an initial configuration where each element forms an individual cluster (*singleton*) and the update steps are cluster merges. This strategy, known as *agglomerative clustering*, elementarily produces a strict hierarchy of clustering configurations. Criteria to assess the best merge to perform are, for instance, the similarity of most similar members of the clusters to be merged, or the similarity of their most dissimilar members (known, respectively, as the *single linkage* and *complete linkage* methods, Duda, Hart & Stork, 2001, pp. 553-554). The stop condition in these examples is either the formation of a pre-specified number of clusters k or a pre-specified similarity threshold value beyond which clusters are not merged any more.

Other feasible types of update steps employed by different methods, other than cluster merges, are: cluster splits, reassignments of a single data element and reassignment of all elements. Identifying the optimal update step, which means quantifying each steps' quality relatively to the other candidate steps, should be carried out with low computational complexity in order to maintain tractability.

The stepwise scheme, though prevalent, is not the only conceivable search strategy. Blatt, Wiseman & Domany (1997), for instance, implement a Monte Carlo procedure producing a series of partitions that can be thought of as “typical” rather than “target” clustering configurations. Then, the probability of element pairs to be part of the same typical cluster is used to determine the final configuration. (Formally, however, this process can also be understood as a conversion – or re-representation, see 2.1.3.3 above – of the given similarities into similarities of other type, namely probabilities to share the same “typical” cluster, followed by application of the single-linkage method with threshold 0.5).

2.1.4.2 Cost-Based Search

There are data clustering methods that assign to each admissible clustering-configuration a measure of quality, or a *cost function*. The availability of global quality measure or cost function has several attractive consequences. The target of data clustering becomes clear and concrete: finding a configuration of highest quality or lowest cost. Accordingly, the stepwise scheme described in the previous subsection can be made more concrete given a cost function, so that the criteria of how to choose the update step to execute next and when to stop the iterative loop are clear and explicit:

```
Given an initial configuration of clusters
repeat
      from a set of currently available update steps of a pre-specified
      type, perform the one reducing a pre-specified cost more than any
      other step, so that a new clustering configuration results
Until the cost cannot be reduced by any available step
```

Figure 2.2: The cost-reduction clustering scheme.

The above iterative loop is guaranteed to stop, as the cost is bounded from below (the number of configurations is finite) and is reduced every iteration. It is apparent, however, that the obtained configuration is not necessarily of absolutely lowest cost, but just one that cannot be improved by the available updates steps. Update step types usually employed are single element re-assignment, picked either according to some fixed schedule or at random (Puzicha, Hofmann & Buhmann, 2000; Slonim, Friedman & Tishby, 2002). In the last case, before terminating the iterative loop, it should be verified that cost cannot improve by any further reassignments, not just of the element currently examined.

2.1.4.3 Axiomatic Approach to Data Clustering

An attractive aspect of cost functions is that they provide a definite criterion, in terms of precise mathematical expression, for determining the quality of data clustering outcome relatively to other possible outcomes. However, in order to make a rough definition of the data clustering problem (such as the one above at the top of Section 2.1) precise, a cost function must involve further assumptions that are not necessarily consensual, but rather they reflect more individual view of what should be expected of "a reasonable clustering procedure". Assumptions of this kind are sometimes incorporated within an axiomatic approach to data clustering. We exemplify below two cases where such assumptions are incorporated in the pairwise clustering setting.

Kleinberg (2003) includes a requirement termed *consistency* in his axiomatic analysis. This requirement states that in any specific clustering problem the best (lowest-cost) clustering

configuration should remain the best solution also for data that is modified relatively to the original problem as follows: some or all of the similarities between elements sharing the same cluster in the best configuration are increased and between-cluster similarities are decreased. Puzicha, Hofmann & Buhmann's (2000) theory of proximity-based clustering (on which we elaborate in Chapter 4; see detailed review there) introduces other requirements. For instance, the relative ranking of all clustering configurations is not expected, under their assumptions, to change in case all pairwise input values are multiplied or shifted by a constant (*scale* and *shift invariance* properties, respectively).

2.1.4.4 Prototypical Representatives of Clusters

Some clustering algorithms keep prototypical vectorial representation of each one of the clusters, additional to the list of cluster members (as mentioned before, such representations can be understood as a part of the algorithm output).

The incremental data clustering scheme (Figure 2.1) can make use of prototypical representations. When they are incorporated, assessing the candidate update steps can rely on associations between elements and cluster representatives, which are fewer than the associations between all pairs of elements involved. On the other hand, the incremental scheme would require in that case updating the cluster representatives along with the updates of assignments into clusters:

```
given an initial array of k cluster representatives
repeat
      - reassign all elements - each element to the cluster with the
        representative to which it is most similar
      - recalculate cluster representatives, based on the members of each
        cluster
until a stable configuration is obtained
```

Figure 2.3: The *k*-representatives clustering scheme.

The *k*-representatives scheme is a specific heuristic variant of the incremental scheme. The specific stopping condition, namely obtaining a stable configuration where cluster representatives are not liable to any further change, is not guaranteed in general terms. It turns out, however, that in some concrete cases, with certain definitions of cluster representatives and element-representative similarity relations, the *k*-representatives scheme happens to reduce a specific cost criterion (known as the *Lyapunov function* of the algorithm, McKay, 2003, p. 299). Therefore, such cases, three of which are mentioned below, realize also the cost-based clustering scheme (Figure 2.2) and are hence assured to stop with configuration that locally minimizes the specific cost.

Feature-based methods naturally represent clusters as vectors in the same space of their input vectorial representations:

- **The *k-means* algorithm**: Each cluster is represented by its *centroid* – the mean of the vectorial representations of the cluster's elements. The distance (dissimilarity) between the centroids and the data elements is measured in L_2 (Euclidean) norm. In this case, the cost being reduced happens to be the sum of squared L_2 distances of all data elements, each from its cluster centroid (Estivill-Castro, 2003).

Duda, Hart & Stork (2001, p. 550) note that, in terms of susceptibility to being trapped in local minima, the *k*-means algorithm has been found empirically advantageous over cost based search with reassignment of a single element at a time.

- **The *k-medoids* algorithm**: Each cluster is represented by its *medoid* – a vector where each entry is the median value of the corresponding cluster-member entries. Element-medoid similarity is captured through proximity in L_1 norm. The cost being reduced in this case is the sum of L_1 distances between all elements and their cluster medoids (Estivill-Castro, 2003).

The notion of cluster representative is applicable, though not very intuitive or common, also in pairwise clustering:

- The prototypical representative is a concrete data element, for which the sum of similarities to all other members of its cluster is the largest compared to corresponding sums of the other cluster's members (equivalently, the sum of dissimilarities is required to be the smallest in the cluster). The cluster-element similarity or dissimilarity employed is straightforwardly given by the proximity measure between data elements. The cost being reduced in this case is the sum of dissimilarities, or minus the sum of similarities, between all the elements and their corresponding cluster representatives. (It is easy to verify that, in each iteration, the total cost is decreased by both reassignments and re-selected representatives).

2.1.4.5 Stochasticity

Randomness can affect several aspects of the algorithmic schemes introduced so far. For instance, the initial configuration can be determined randomly and so is the next element to be reassigned, in case a single element is reassigned at each update step of the incremental scheme. More importantly, update steps can be chosen stochastically among all alternatives as the following scheme explicates:

```
given an initial configuration of clusters
repeat
      pick at random one of the currently available update steps s and
      perform it with probability correlated with the anticipated addition
      to cost Δs
until, for some pre-specified number of iterations, the change in cost
does not exceed some pre-specified small threshold value
```

Figure 2.4: The stochastic cost-driven clustering scheme.

There are a couple of algorithmic variations that fit this scheme. They differ from one another by the way they calculate the probability to perform a candidate step as a function of its anticipated impact on the cost. As with the basic cost-reduction scheme, the update steps in both variations are picked from all possible reassignments of one randomly chosen element. In one variation of *simulated annealing* (Kirkpatrick, Gelatt & Vecchi, 1983) if an update step that does not increase the cost is picked it would be performed always (i.e. in probability 1). A step s that does increase the cost by a positive quantity Δs would be performed in probability $e^{-\beta \Delta s}$. In another variation (*Gibbs sampling*, Geman & Geman, 1984; implemented in Chapter 4) any step s can be picked and performed in probability proportional to $e^{-\beta \Delta s}$, where the cost change Δs can be either positive or not. In both cases, the cost might occasionally grow, though rarely in comparison to cost reduction. Stochasticity is gradually relaxed during execution of both variations by means of gradual increasing of the inverse "computational temperature" parameter β. Initially, β is low, which implies low differentiation between varying levels of cost-change. During execution β is gradually increased, so that after a large number of iterations a high β value is employed and systematic increase in cost becomes very probable even in comparison to slight deterioration.

Theoretically, and often in practice, in order to ensure that the algorithms above produce a clustering configuration of globally low cost, a very slow schedule of stochasticity relaxation (i.e., very gradual increase of the β parameter) is required, which results in long execution time (Duda, Hart & Stork, 2001, p. 356).

2.1.4.6 Probabilistic Clustering

As mentioned before (in Subsection 2.1.2.2), there are methods that compute non-negative probabilities of assignments or "assignment levels" $p(c|x)$, which, per element x, sum up to one over all clusters c. We will now refer to those methods assuming further that the given feature-based vectorial representations of the elements are also normalized: each element x is represented by conditional probability distribution $p(y|x)$ over the features. Given such normalized representation of the data, it is natural that cluster representatives are taken from the same probabilistic space: each

cluster is represented by a conditional probability distribution over the features, $p(y|c)$. This probabilistic setting underlies the following scheme, which generalizes the *k*-representatives clustering scheme of Figure 2.3:

---

given initial assignment levels $p(c|x)$ for each element x and cluster c,

repeat

- recalculate each cluster probabilistic representative $p(y|c)$, as a normalized sum of all element probabilistic representations $p(y|x)$, weighing the contribution of each element x by its assignment probability to c, $p(c|x)$.
- recalculate each assignment probability $p(c|x)$ for each element x and cluster c, so that it is proportional to the similarity of x's vectorial representation, $p(y|x)$, to c's vectorial representation, $p(y|c)$.

until the recalculated representatives do not change any more beyond some pre-specified small threshold value

---

Figure 2.5: probabilistic representative-based clustering scheme.

We refer below to two approaches that fit in the above probabilistic representative-based scheme. These two approaches, whose underlying algorithms are very similar to one other, are related with the two motivations mentioned earlier for the data clustering task: "prediction" versus "communication" (Subsection 2.1.1). One approach – the one associated with the predictive aspect of data clustering – follows the *expectation maximization* (EM; Dempster, Laird & Rubin, 1977) framework. It examines the data as if it were sampled from a mixture of latent distinguishable classes of element-feature co-occurrence distributions to be approximated. The iterative steps of calculating assignment probabilities $p(c|x)$ and cluster representatives $p(y|c)$ maximize a likelihood term of the mixture model. Deriving the update steps so that they systematically maximize the model likelihood implies a particular representative-element similarity measure,

$$sim_{EM}(x, c) = p(c)\, e^{-count(x)\, KL[\, p(y|x) \,\|\, p(y|c)\,]}, \qquad (2.7)$$

where $p(c)$ is the relative weight of cluster c and $count(x)$ is the total number of occurrences of the element x in the data. A particular work that is based on this formulation is Hofmann, Puzicha & Jordan's (1999) *one-sided clustering model*. As the data – i.e., the number of times each element is sampled – grows, the method is capable of assigning the elements more deterministically and detecting larger numbers of clusters.

The second approach is related with the communication aspect of clustering and is based on information theoretical considerations. The *information bottleneck* method (IB; Tishby, Pereira & Bialek, 1999) implements this approach. It looks at data clustering as if it aims at lossy encoding of

the data so that relevant information, namely information about the features, is conveyed optimally. In this case as well, a similarity measure emerges from the principles underlying the IB method, which turns to be very similar to the EM related similarity measure:

$$sim_{IB}(x, c) = p(c)\, e^{-\beta KL[p(y|x) \,\|\, p(y|c)]}, \qquad (2.8)$$

where $p(c)$ is, again, the relative weight of the cluster c and β is an additional parameter, which, roughly speaking, articulates counterbalance between the target of communicating the information as accurately as possible and the target of reducing the length of communicated transmission. In Chapter 5, we provide more detailed description of the IB method and the closely related *information distortion* method (Gedeon, Parker & Dimitrov, 2003; Pereira, Tishby & Lee, 1993).

Further examples of methods following the probabilistic representative-based clustering scheme can be seen as close variants on the methods mentioned above, for instance, the three *soft k-means* versions that are specified by MacKay (2003, Ch. 20, 22). Another method that is worth mentioning in this context is Bezdek's (1981) *fuzzy c-means*, which does not require the assignment levels of an element to sum up over all clusters to one, but leaves space for uncertainty regarding the assignments.

2.1.5 Variations on Data-Clustering

We now discuss some methods that are related to, or extend, the data clustering task.

2.1.5.1 Data Clustering and other Unsupervised Tasks

Data clustering is a key member in the family of unsupervised computational learning methods. It is interesting to note that data clustering can be seen as if it accomplishes tasks that we have already mentioned as means for pre-processing prior to clustering (Subsection 2.1.3.4).

- **Similarity assessment**: any standard deterministic data-clustering configuration imposes a trivial 0/1 similarity measure: elements of the same clusters are considered similar, while elements of different clusters are not. Probabilistic clustering is more detailed in this respect: assignment probability distributions of individual elements over the clusters, $p(c|x)$, can be used to measure distance or similarity between data elements through standard distance or similarity measures of distributions (e.g., *KL* distance). This might facilitate overcoming the sparseness that often characterizes raw element-feature co-occurrence vectors in domains such as text processing.
- **Dimensional reduction / feature generation**: clustering is also a particular case of dimensional reduction. Specifically, probabilistic clustering maps the data onto a k-1 dimensional simplex embedded in a k dimensional space, where k is the number of clusters. The utility of dimensional reduction of the feature space – replacing the original array of features by clusters of features – is demonstrated by Slonim & Tishby, 2000.

2.1.5.2 Methods that Extend Basic Data Clustering

There are data clustering methods, both deterministic and probabilistic, that process and output constructs more elaborated than plain clustering configurations. There are several methods that output a *hierarchy* of clusters, i.e., a sequence of increasingly detailed clustering configurations consisting of an increasing number of smaller clusters, so that every cluster forms a subset of a cluster in each one of all less detailed configurations. Simple examples are the complete and single linkage methods mentioned before (Subsection 2.1.4.1). Hierarchical clustering, however, can be tackled through more sophisticated approach, including the case of probabilistic clustering (Hofmann, Puzicha & Jordan, 1999). Although in the current work we do not aim at hierarchy in general, we note that in general meaningful sub-clusters that emerge as individual clusters in a more detailed configuration are often revealed simply by applying a non-hierarchical method repeatedly with number of clusters, k, incremented at each run. Such partial hierarchy often reveals interesting relations between themes and sub-themes in the data (as demonstrated in Chapter 4 and Chapter 5).

One further well known modification to the data-clustering task is the *self-organizing map* (SOM; Kohonen, 1989) that, in addition to a clustering configuration, maps the clusters onto a grid, thus imposes spatial structure on the clusters.

A recent line of works on dimensionality reduction (the *two-sided clustering model* by Hofmann, Puzicha & Jordan, 1999; Hofmann, 1999; Globerzon & Tishby, 2002), can be seen as extending feature-based clustering to a setting of two sets of elements, symmetrically playing the roles of both clustered data and features with respect to each other. A further recent work, on the *multivariate information bottleneck* method (Friedman et al., 2002), extends the concept and technique of the IB method in revealing complex relational constructs. The input to this method may consist of several distinct element sets that are connected with one another through a network of element-feature relations. The method can produce several clustering configurations, each of which partitions (probabilistically) one of the sets that take the role of clustered data. The obtained complex relational structure allows, for example, several different partitions of the same set simultaneously, directed by different relevance feature sets conveying different types of information regarding the "multi-clustered" set.

2.1.5.3 Data Clustering with Constraints

There are several works, including the present one, on methods producing output that has the form of a standard clustering configuration, but differ from ordinary data clustering by considering input that is supplementary to the standard array of pairwise proximity values or feature-based vectorial representations.

In a review of relational data-clustering methods used within social sciences, Batagelj & Ferligoj (2000) provide examples of combining clustering with relational biases of various kinds that are embedded within the data. The *blockmodeling* method seeks to cluster together elements that have similar patterns of relations with other elements (in this case, some representation of the global relationships between clusters may form supplementary output). There are several types of relational pattern similarity, such as *structural equivalence*, where elements are identically related with the rest of individual elements, and *regular equivalence*, where elements are similarly connected to equivalent other elements. Another approach reviewed by Batagelj & Ferligoj (2000) – *constrained clustering* – groups similar elements into clusters based on features, but clusters have to satisfy also some additional conditions. For example: clusters of geographical regions that are similar according to their socioeconomic development level have to be determined such that the regions inside each cluster are also geographically connected. Considerations that, in a way, are similar to the above are applied in works on image segmentation, where nearby pixels are relatively probable to be part of the same object (Boykov, Vexler & Zabih, 1999).

Other works (Wagstaff et al. 2001; Basu, Banerjee & Mooney, 2002) tackle data clustering constrained by other types of pre-specified restrictions. They take as an additional input a list of element pairs constrained to be in the same cluster, versus another list of pairs constrained to be in different clusters. Several variants of the k-means algorithm that incorporate such constraints were suggested.

The method of *information bottleneck with side information* (Chechik & Tishby, 2003) can be viewed as extending further the above line. Chechik & Tishby consider an additional set of features conveying "negative" information that is supposed to be neutralized rather than be followed in forming the clusters. Our work addresses a closely related task, so we will provide detailed comparison with this work.

2.2 Computational Models of Analogy

Analogy is defined as "similarity in some respects between things that are otherwise dissimilar" (quoting http://dictionary.reference.com/search?q=analogy). The issues of how analogies are identified and what makes an analogy a good one have been discussed in the cognitive literature from a variety of points of view. A major motivation to studying computational methods for identifying analogies (*analogy making* in short) is that the capacity of drawing analogies and metaphors is an essential part of human intelligence. Analogy making allows utilizing knowledge, ideas and inspiration across seemingly different domains and thus it contributes significantly to the flexibility and creativity characterizing human intelligence. "*Analogy pervades all our thinking, our everyday speech and our trivial conclusions as well as artistic ways of expression and the highest scientific achievements"* (Polya, 1957).

In addition to the theoretic motivation of modeling an essential ingredient of human intelligence, developments in the computational methods used for analogy making – including, we hope, the ones introduced in this work – might have impact on practical applications: language understanding and generation, data mining, artificial intelligence and so on.

Our review below is only exemplifying. We concentrate on two approaches to analogy making: the *structure mapping theory* (Gentner, 1983) and the *Copycat* project (Hofstadter et al., 1995, Chapters 5-6). Among the many works to which our review does not refer, there are several that can be seen as lying somewhere in between the two above mentioned ones (e.g., Holyoak & Thagard, 1989; Hummel & Holyoak, 1997; Veale, O'Donoghue & Keane, 1999; Kokinov & Petrov 2001)[1]. More comprehensive discussion with reference to a larger variety of works is given, e.g., by French (2002).

2.2.1 The Structure Mapping Theory

As French (2002) notes, Gentner's structure mapping theory (SMT; 1983) is unquestionably the most influential work to date on the modeling of analogy-making. It has been applied in a wide range of contexts ranging from child development to folk physics. The prominent innovation of SMT relatively to earlier works is the emphasis that it puts on structural similarity between the analogized

[1] All in all, these works process data consisting of relational prepositions similar to the representation used by the computational implementation of the structure mapping theory (see Section 2.2.1), but they employ computational machinery, such as connectionist or neural network architectures, somewhat closer in spirit to Copycat (2.2.2).

systems. The *structure-mapping engine* (SME; Falkenhainer, Forbus & Gentner, 1989) is the computational implementation of SMT.

2.2.1.1 Data Representation

SME represents the information about the systems between which analogy is to be drawn as relational prepositions. Unary relations (of one argument) are equivalent with elementary attributes or features, as is familiar from the conventional feature-based data-clustering setting. A numeric value might be used to quantify the association between an element and its attribute. For example (based on Falkenhainer, Forbus & Gentner, 1989), the fact that 'temperature' is an attribute of coffee, with value *X*, is denoted as:

`TEMPERATURE(coffee) =` *X*

In distinction from elementary features, there are relations that apply to two or more elements and even to other relations, e.g.,

`GREATER-THAN [ TEMPERATURE(coffee), TEMPERATURE(ice-cube) ]`

or to any fixed combination of relations and data elements.

2.2.1.2 Principles and Algorithmic Framework

Two major principles underlie SMT:

- the relation-matching principle: good analogies are grounded on mapping of multi-argument relations rather than attributes (unary relations).
- the systematicity principle: mappings of coherent systems of relations (i.e., graphs resulting from compositions of relations) are preferred over mappings of individual relations.

The SME algorithm implements a heuristic search for a cross-system map realizing these principles through four stages:

- Local match construction: find all base-target element pairs that can potentially match, i.e. all possible matches between groups of elements sharing *identical* relations in base and target systems.
- Global map construction: within the formed collection of all possible local matches, identify all maximal consistent sub-collections of matches.
- Candidate inference construction: for each maximal consistent sub-collection of matches, infer additional relations not given originally in the target domain that have

matched relations in the base domain and thus extend the map suggested by the consistent matches.

- Match evaluation: calculate a score for each one of the candidate extended maps incorporating the inferred information, based on local structural measures that are derived from the two SMT principles above.

Although innovative and influential (and maybe because of it), SMT has been extensively criticized, for example by Hofstadter et al. (1995, mainly Ch. 4). Hofstadter et al.'s criticism is particularly focused on the inflexibility inherent to SME. This inflexibility is expressed in the one-to-one mapping scheme that is restricted to mapping of identical – even not similar – relations. Further, the propositional representations that the program manipulates are manually coded and, as such, they articulate pre-determined relations over a pre-determined set of concepts. The SMT pretends to account for real-world analogies. However, it is not clear to what extent it models successfully the flexibility and creativity characterizing human reasoning, particularly if one has in mind *real-world data* in raw form that cannot be suspected as tailored for the problem at hand.

2.2.2 The Copycat Project

The Copycat project by Hofstadter et al. (1995; Chs. 5-6) is one of several projects from the same group (see other chapters there), promoting an approach alternative to the seeming rigidity of SMT. A basic strategic direction of Hofstadter et al. is abandoning the pretension to solve real-world problems of general character, which in their view is a too advanced challenge for the present level of recent research. Rather, they advocate concentrating on specific artificial toy domains. Simple toy problems are claimed to underlie search spaces that are more liable to systematic study (as Hofstadter et al. put it: looking at a problem together with its "hallo" of variant problems; p. 330).

2.2.2.1 System Overall Description

Copycat is a computer program exemplifying the above direction. It is restrictedly designed to answer questions regarding analogy of letter strings transformation, such as:

> "if a string ***abc*** is transformed into ***abd***, what would be the analogously transformed value of a target string ***xyz***".

The strings ***abc*** and ***abd*** form the *base* domain of the problem and the ***xyz*** string is the given part of the *target* domain. The problem to be solved is completing the target domain by constructing an additional string so the relation between the two target strings, the given one and the constructed one, will be analogous to the relation between the two base strings.

The solution is achieved based on grouping letter subsets together and on two types of links between the original letters and between formed letter groups: *bonds* that link neighboring elements within the same string and *bridges* that connect (map) between different strings. Thus, a possible solution to the problem presented above as an example can be that ***xyz*** is transformed into ***wyz***. This solution might rely, among other things, on bridging the letters *c* and *d* (across the two base strings) and a corresponding bridge between *x* of the given target string and *w* of the constructed solution string.

The program considers rather elementary information regarding the nature and characteristics of the alphabet letters. *a* is recognized as the first letter, and *z* is recognized as the last one. The alphabetic order, i.e. the identities of the letters that come before and after each letter (successor and predecessor relations), is known as well. *Slipnet* is the name of a central ingredient in the Copycat architecture, which stores and process this information in addition to other information that is gathered during the run. It consists of approximately 60 nodes. The values associated with the Slipnet nodes form a vector, referring globally, across the whole system, to the intensity of both basic features and relations as the ones mentioned (*first letter*, *last letter*, *successor*, *predecessor*, and also *the letter A*, *the letter B*, …, *the letter Z*) and, additionally, features and relations that are not specified in advance but rather emerge during Copycat's execution. Examples for these emerging attributes (*meta-features*) are the notion of *LETTER, SUBSEQUENCE LENGTH*, *INCREASING SEQUENCE* (and *DECREASING SEQUENCE*) and *OPPOSITE*.

In order to concretize further the framework described above, we draw the following simplistic partial example. Suppose that one of the base strings and the given target string are ***aab*** and ***bba*** and ignore for the moment the other details of the Copycat setting (which are not much relevant in terms of our current work). Suppose further that the following grouping pattern has been evolved: in ***aab*** the first two letters are grouped together so that the whole string is now perceived as concatenation of ***aa*** and ***b***. Similarly, ***bba*** is perceived as a ***bb*** group concatenated with ***a***. This grouping pattern is not guaranteed to emerge in a real run, but seems probable in view of actual test cases described by Hofstadter et al. The ***aa*** and ***bb*** groups are likely to be marked by the features *letter A* and *letter B*, respectively, as well as by the feature *length 2*. The solution where ***aa*** is bridged to ***bb*** and ***b*** is bridged to ***a*** would have an increasing impact on the Slipnet value associated with the feature *SUBSEQUENCE LENGTH*, reflecting that the matched groups share the same length. An alternative conceivable solution, where ***aa*** is matched with ***a*** while ***b*** is matched with ***bb*** would result in increase in the value associated with the features *LETTER*, reflecting that the matched groups consist of the same letter, and *OPPOSITE*, reflecting that the groups are matched in their reverse order.

To summarize, the computational machinery underlying Copycat is based on three main factors that constantly change, while interdependently affecting each other. These are the Slipnet global values, the current setting of letter grouping, bonds and bridges, and the purposed solution string, which is firstly constructed after some execution time but from this point on, is also constantly adapted to fit the current state of the other factors and at the same time affects them.

2.2.2.2 Further Discussion in View of Methods Reviewed Previously

The computational framework of Copycat is somewhat reminiscent of the stochastic data clustering methods (Subsection 2.1.4.5), and of the scheme of constantly adjusted assignments and gradually stabilizing probabilistic or deterministic centroids (2.1.4.6 and 2.1.4.4). Update steps – grouping and ungrouping, setting and unsetting of bonds and bridges, attaching labels (features) to groups, adapting Slipnet values, and so on – are stochastically chosen from a pool of prioritized *codeletes*: small code segment that are randomly chosen to be performed. In difference from some of the methods we described, choosing the next step, i.e. codelete, to perform is not based on a global one-valued cost criterion but rather on a variety of global and local considerations (that are assessed by some codeletes dedicated for this purpose).

Among the various considerations being constantly assessed, there is a global computational-temperature parameter, which is described as regulating a global level of "open-mindedness" expressed through stochasticity level and overall likelihood of certain types of codeletes to perform. In relation to that, the temperature quantifies a global "confidence" level with regard to the currently posed solution, so that once it goes below a certain level the probability of terminating the run increases. The temperature has no deterministic cooling schedule as in simulated annealing (2.1.4.5) or in the IB method (described in Chapter 5).

Evolving representation, which is not hand coded or pre-determined, seems to be a unique and fascinating aspect of the Copycat project. On the other hand, Copycat employs a complex and hard-to-analyze computational mechanism and at the same time it manifestly gives up addressing practical real-world problems. The data-clustering-based methods we introduce later in this work (Chapter 4 and Chapter 5) attempt to maintain the flavor of creative gradually-emerging analogy discovery, along with fairly tractable computational rational and mechanisms and implementation to real-world data.