Metrics and visualisation for crime analysis and genomics

Laros, J.F.J.

Chapter 5

Temporal Extrapolation Using the Particle Model

In the field of criminal investigations people on the task force usually have a gut feeling about trends in criminal careers. For example, it is reasonable to assume that a drug addict that has been doing petty theft and some other minor criminal offences, combined with drug abuse, will keep doing to do this in the future. On the other hand, someone of whom we see an incline in the severity of the crimes, has a high probability of committing even worse crimes in the future.

In this chapter we will focus on a way to find similar careers and perhaps to automatically make a prediction of a future path of a criminal career by looking at the trends in the neighbouring criminal careers.

5.1 Introduction

Under the assumption that criminals that have a similar history, will have a similar future (at least in the near future), we propose an analysis based upon a large database (obtained from the Dutch national police, which for privacy reasons can not be disclosed) which consists of information on approximately one million criminals and their criminal history. We use a dimension reduction technique for noise reduction and to simplify the calculations. We use metrics designed especially for criminal careers as a basis for this dimension reduction.

In earlier work [14] temporal extrapolation was successfully attempted within a static clustering. In this work, criminals were classified by progressively adding their career to the static clustering. In this way, the first year resulted in a position in the clustering, the first two years were an other point and so on. The next point is determined by a 2-dimensional parametric extrapolation.

A disadvantage of this technique is that it relies on the linearity of the dimension reduction technique. If the produced image is done with a non-linear technique, the result may be less usable, depending on the input data.

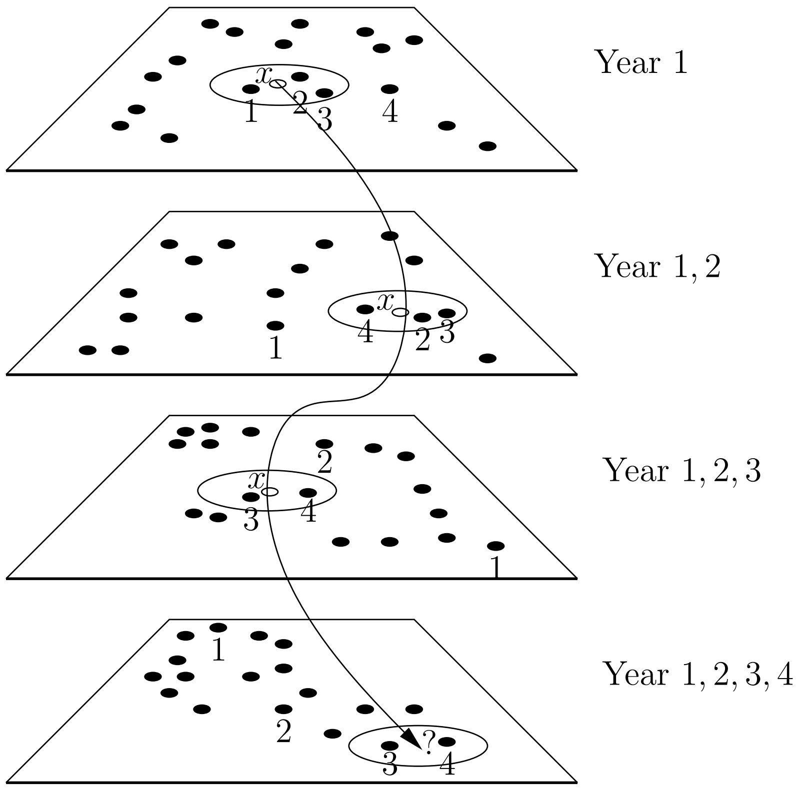

Figure 5.1: Flowing model

To accommodate for the shortcomings of the previous extrapolation algorithm, we use the idea of progressively adding a career to all points, instead of one. By doing this, we create a dynamical model that changes through time as the careers of the data points progress. By keeping track of the neighbourhood of the target point, we can extrapolate the end position of the target without making any assumptions about the space at all.

In Figure 5.1 we see an example. In the upper plane, only the first year of each career is considered, in the second plane the first two and so on. In each plane we see the tracked point (an open circle) denoted by x, and a couple of points that are or have been in the neighbourhood of x, they are denoted by 1, 2, 3 and 4.

We decide upon a neighbourhood in some way and keep track of the careers in this neighbourhood throughout the years. In the last step, we either use the neighbouring careers as such and return them for further analysis, or we can look at the subsequent crimes in all of these careers and by employing a weighing scheme, do a prediction for future crimes.

The extrapolation can be done in two ways: We can either do a new dimension reduction for each added year, or we can alter the input matrix to add a new year. If we would only do one extrapolation per target point, and if a non-deterministic dimension reduction technique is used, the latter choice might be the best one, because if the dimension reduction technique gets stuck in a local optimum, the on-line adding of new data would use this optimum as a starting point. Running multiple extrapolations though, is better in both cases and obsoletes the on-line option.

We use a *particle*- or *push and pull* model (see Chapter 2) for the dimension reduction step, because it is fast and has flexibilities that other techniques lack, such as the choice of an output space (see Chapter 6. We use a *torus* as output space in this chapter because we are only interested in the (local) neighbourhoods of each point, which the torus preserves. Furthermore, the fact that the torus is a closed surface gives more freedom for a valid visualisation.

5.2 Parameters

There are a number parameters in this approach. We shall discuss the parameters of each step in detail.

Generating the distance matrices First of all, we make a number of distance matrices (one for each year as the career progresses). As a distance measure we use the *edit distance* between two multiset sequences (see Chapter 7, each of these multisets represents the activities of one person in one year. This step itself already has a large amount of parameters. We can either use *local* or *global alignment*, we can use *expanded* or *non-expanded* careers and we have to decide on a *gap penalty* and a *edit penalty*. These parameters are discussed in more detail in Chapter 7. All of these parameters should be decided upon by a domain expert.

Dimension reduction When the distance matrices have been generated, we do a dimension reduction step. We mainly do this for *noise reduction*, but it also has the nice side effect that it makes the rest of the calculations less demanding. After this dimension reduction step, we need to decide upon the size of the neighbourhood and how to determine it. One could for example use a fixed radius with the tracked point as a centre, or we could use k nearest neighbours.

Since we use a model that can produce non-linear mappings onto the output space, we can not make the neighbourhood too large, we can not make it too small either, because it might be empty then. These two observations make the choice for the radius a difficult one, and even dependent on the input data, or sometimes even on the tracked point.

The k nearest neighbours approach solves most of these problems, but the non-linearity issue is still bothersome. Also, we obviously need to assign a sensible value for k.

We propose a hybrid solution for the non-linearity issue. We choose k nearest neighbours, but we also record their distance to the tracked point. This way we can use this distance as a weighing function in the extrapolation by using $1/distance$ as a weight for example.

Weighing of the neighbouring careers Finally, we need to come up with

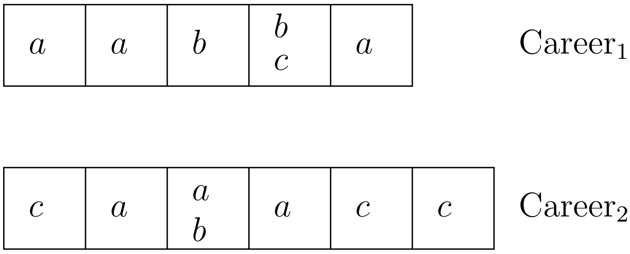

Figure 5.2: Two careers

weights that accentuate the difference between neighbouring points in the first year, neighbouring points in the first two years and so on. We do this because we assume that similarity in the last year is more significant than similarity in the first year.

Prediction As an extra step, we can use the neighbouring careers as a predictor for the future actions of the person corresponding to the tracked point. Again, there are multiple choices to consider. We can for example only use the data from the next year as a predictor, we could use multiple years, or we can take the remaining career of each neighbouring career to make a prediction. In the latter case, we propose a weighing scheme that assigns high weights for crimes in the near future and low weights for those in the distant future.

5.3 Extrapolation method

To do the extrapolation, we first need to calculate the distance matrices for the progressing careers. Suppose we have n objects and we want to do the extrapolation over m years, We first have to make the matrix for the first year, then the one for the first two, and so on. This pre-calculation step results in m symmetric matrices of $n \times n$ distances. We denote the matrix which consists of the distances for the first year only by M_1, the one for the first two years be M_2 and so on.

In Figure 5.2, we see two careers. To generate M_1, we take a look at the first field of each career, to generate M_2, we take a look at the first two fields. The distance measure we use for this comparison is the edit distance of multiset sequences (see Chapter 7).

We start the extrapolation by making a dimension reduction of the points corresponding to the distances in M_1 and pinpointing the career c we want to follow. We choose a radius, which defines the neighbourhood of the career and then we list all the points in this neighbourhood. We call this set $N_1 = N_1(c)$.

Then we either alter the distance matrix M_1 by replacing the values with the ones in M_2 and not reinitialising the visualisation for the reasons mentioned above (to avoid getting into a different local or global optimum), or we make a new visualisation. Either way, the points in the visualisation will adjust to

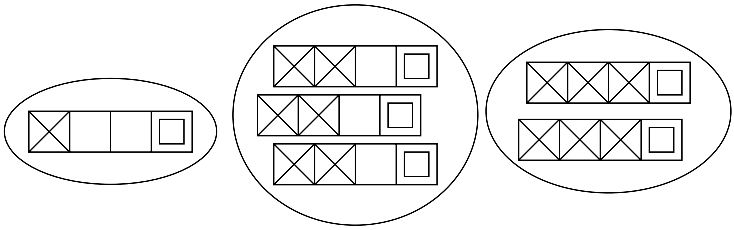

Figure 5.3: Neighbourhood sets for $m = 4$

the new data (M_2) and if the visualisation reaches a stable state again, we can extract the new neighbourhood set $N_2 = N_2(c)$ in the same way as we did above. We repeat this process until we have all $m - 1$ neighbourhood sets.

In Figure 5.3 we see a number of neighbourhood sets. The elements of the careers that are considered are marked with a cross. So in N_1 only the first element of each career is considered. In N_2 the first two elements are considered and so on. Each m-th element is marked with a square to denote the element that is used for the extrapolation.

After this step, we can have a look at element m in each career in the neighbourhood sets. It is reasonable to use a weighing scheme (where N_i has weight w_i), which assigns a large weight to careers in N_{m-1} and a small one to the ones in N_1, to emphasise the temporal aspect of the extracted data.

The weight for each neighbouring career x is calculated as follows:

$$w(x) = \sum_{i=1}^{m-1} \sum_{i:x \in N_i} w_i$$

We can now apply this calculated weight to the m-th element of career x and by doing this for all neighbouring careers, we get a multiset of weighed activities. The weight can be used as an indicator of chance. The outline of the algorithm is as follows:

Predict (c) ::
  for $i = 1$ **to** $m - 1$ **do**
    compute M_i
    do a dimension reduction for M_i
    determine $N_i = N_i(c)$
  for $x \in \cup_{i=1}^{m-1} N_i$ **do**
    compute $w(x)$
  compute prediction using $w(x)$

Prediction for a career c based upon neighbouring careers

5.4 Experiments

In our experiments, we took a random selection of 112,326 criminals from our input dataset (consisting of the criminal activities of Dutch offenders) and applied a filter to select those people that had a career of at least four years. This resulted in a new database consisting of 1,617 criminal careers.

The crimes (elements of the multisets) were categorised in 8 categories, which the Dutch National Police themselves use and therefore seem reasonable. These categories have not been weighed in our experiments, although some categories are representative for minor offences, while others are representative for severe crimes. We have chosen not to weigh crimes, because we want to find a prediction in more detailed behaviour, while predicting the nature of future crimes would require this weighing scheme.

For the construction of the distance matrices, we used local alignment as a scheme to calculate the edit distance. The gap penalty was set to the maximum edit penalty, which was 1.0. Note that because we search for the edit distance between strings of the same length (in every step, we only consider the first n years of a career that is longer than n), there is no difference between local and global alignment.

In the dimension reduction step, we used a fixed number of neighbours $k = 10$ and used the hybrid weighing function to compensate for "neighbours" that are far away. We made a dimension reduction for three years and recorded all the neighbouring careers for each step. Each dimension reduction was repeated four times to compensate for the possibility of local optima.

Year	1	1–2	1–3
Value	1	4	16

Table 5.1: Weights used for extrapolation

The weights for careers found in each step is given in Table 5.1. These weights were in turn divided by the distance from the tracked point, as explained in Section 5.2. In this set up, we predict the future crime with 63% accuracy, vs. the 11.1% we would get if a random result would have been returned.

5.5 Conclusions and further research

Because the dimension reduction technique was not used for the normal 2-D visualisation, but as a noise filter only, we can use a dimension reduction technique that reduces the input space to a higher dimensional space than the standard 2-D. In this way, the embedding of the calculated distances will probably be a lot better, but in general we shall need a larger radius to find a sufficient amount of neighbours in order to do the extrapolation. In other words, the accuracy will improve (up to a certain point, we still need the noise reduction of course), but at a cost. The computational complexity of the problem will increase severely.