Matching of broken random samples with a recurrent neural network

Rudolf Frühwirth

Institut für Hochenergiephysik der
Österreichischen Akademie der Wissenschaften
Nikolsdorfer Gasse 18, A-1050 Wien, Austria
E-mail: fruhwirth@hephy.oeaw.ac.at

To appear in *Nuclear Instruments and Methods in Physics Research A*

Abstract

Various problems arising in the reconstruction of high-energy physics events can be solved in the general framework of matching of broken random samples. Typical examples are signal matching, track-track matching, and track-hit matching. If the random samples are complete and free of noise, there exists an optimal statistical procedure. In a real-world application, however, the sample usually is noisy and/or incomplete due to observation inefficiency. In this case the optimal procedure is inapplicable. We investigate a sequential and a global approach to solving the resulting combinatorial problem. The global approach is implemented by a recurrent neural network with mean-field annealing. We analyze the performance of both methods for various types of random samples and various levels of inefficiency and noise.

1 Introduction

500 9447

Various problems which arise in the course of the reconstruction of high-energy physics events can be considered as special cases of a general problem, namely the matching of broken random samples. We list a few examples of such problems.

- **Signal matching.** In a wire chamber with two-dimensional read-out signals induced by charged particles are measured via two different channels, anode and cathode. Different types of measurements are possible, e.g. time of arrival, pulse-height, or integrated charge. In order to gain information on the position of the particles which generate the signals, it is necessary to match anode and cathode signals by comparing the measurements in the respective channels. It cannot be expected that the channels are free of noise and fully efficient.

- **Track-track matching.** Suppose that a set of tracks has been measured in two independent subsystems of the detector. If the information from both subsystems is to be combined, an optimal matching has to be found. Again, as a rule, tracks may be missing in one of the subdetectors, and ghost tracks may have been found, which can be considered as noise in this context.
- **Track-hit matching.** Suppose that a set of tracks has been extrapolated into a subdetector which is incapable of measuring a full 5-parameter track element, for instance a microstrip vertex detector. It is of the utmost importance to get the best matching of extrapolated tracks and measurements (“hits”) in the subdetector. Again, not all tracks may be present in the set of extrapolations, and the subdetector is usually not fully efficient or noisy.

It should be noted that in all cases the components of the broken sample are measured on the same scale, possibly after a suitable calibration. Consequently there is a natural measure of distance between any pair of observations (x_i, y_j) . The matching procedures we are going to investigate are based on this concept.

In the simplest matching problem, a random sample of size n is drawn from a bivariate population. What is observed, however, are not pairs $(x_i, y_i), i = 1, \dots, n$, but only components $x_i, i = 1, \dots, n$ and $y_i, i = 1, \dots, n$. The aim is to match the observed values in order to reproduce the pairs in the original sample. This problem of matching or re-pairing a broken random sample has been considered from a statistical point of view in a couple of papers [1, 2, 3]. It has been shown in [2] that under certain conditions the maximum likelihood pairing (MLP) is given by the “natural pairing”, which matches x_i with y_i after ordering both components of the broken sample. The MLP maximizes the probability of pairing correctly all n observations, and, under certain conditions, also maximizes the expected number of correct matches. It should be noted that the best pairing depends only on the ordering of each component of the broken sample; it is not mandatory that x and y are measured on the same scale. It is, however, required that the sample is complete, i.e. that all observations are available, and that there is no contamination by noise.

In practice, however, this is hardly ever the case, and consequently MLP cannot be used, if only because the number of observations of the two components is different due to inefficiency or noise. Hence, the problem has a combinatorial aspect which leads us to expect that near-optimal solutions can be found by a neural network type algorithm.

We first consider the problem of matching scalar observations measured on the same scale. It arises in the matching of signals according to time of arrival, pulse-height, or integrated charge, and in the matching of track extrapolations with one-dimensional position measurements, for example in a semiconductor microstrip detector. This special situation can be described by means of the following stochastic model of the bivariate population:

$$\begin{aligned} x &= t + \delta_x, & \delta_x &\sim \mathcal{N}(0, \sigma_x^2), \\ y &= t + \delta_y, & \delta_y &\sim \mathcal{N}(0, \sigma_y^2), \end{aligned}$$

where the distribution of t is unspecified, and δ_x and δ_y are assumed to be independent. If t is normally distributed, then so are x and y . Obviously x and y are correlated with

correlation coefficient

$$\rho = \frac{\tau^2}{\sqrt{(\tau^2 + \sigma_x^2)(\tau^2 + \sigma_y^2)}},$$

where $\tau^2 = \text{var}(t)$. The sample of observations is obtained by breaking the pairs (x, y) and by randomly accepting $x(y)$ with probability $\varepsilon_x(\varepsilon_y)$. In addition, both components can be contaminated by noise, i.e. additional observations without matching partner. The actual size of the broken random sample will be denoted as n_x and n_y , the number of pairs remaining after applying inefficiency as n_2 .

A natural measure of distance is given by the weighted squared difference:

$$d_{ij} = (x_i - y_j)^2 / (\sigma_x^2 + \sigma_y^2).$$

If x_i and y_j belong to the same pair, d_{ij} is χ^2 -distributed with one degree of freedom. Clearly, the concept of natural distance can be generalized in a straightforward way to the case where t , x , and y are random vectors rather than scalars, necessarily of the same dimension. If the covariance matrices of δ_x and δ_y are known, the distance function is defined as:

$$d_{ij} = (x_i - y_j)^T (\text{cov}(\delta_x) + \text{cov}(\delta_y))^{-1} (x_i - y_j).$$

This model is relevant for the matching of track segments or of track extrapolations with full 5-parameter track elements.

We can expect to get a good match if the sum of distances between the partners of all pairs is minimized, leading to a least-squares matching. This in turn suggests a suboptimal, but simple sequential algorithm which, unlike MLP, can be applied to incomplete or noisy samples. We call it the “Sequential Least-Squares Matching” (SLSM) algorithm. It runs like follows: Pick the two unused observations with the smallest distance; mark both observations as used; repeat until the observations are exhausted.

The SLSM algorithm could be called a “local” approach; it does not necessarily lead to the global minimum of the objective function. It therefore seems worthwhile to look for an algorithm which is capable of finding the global optimum. It is well known that recurrent neural networks with annealing are well suited to this type of optimization problem, as is exemplified by the solutions to the traveling salesman problem [4], to the problem of finding a maximal independent subgraph of a undirected graph [5], to the graph bisection problem [6], and many others. The architecture of the proposed network is described in section 2. The performance of the network and of the sequential algorithm will be compared in section 3.

2 Architecture of the network

The neural network (NN) which we are going to set up in this section is a Hopfield network with binary neurons [7] and mean field annealing [8]. It is inspired by the networks used to find maximal independent subgraphs [5, 9] and to solve the graph bisection problem [6]. In principle, a binary neuron has to be created for each possible pairing (x_i, y_j) , resulting in $n_x n_y$ neurons. In order to cut down the number of neurons required, we do not create

neurons with large associated distances d_{ij} . If σ_x^2 and σ_y^2 are known sufficiently well, the upper bound $d_{\max}$ can be determined by the quantiles of the χ^2 -distribution, and the losses of correct matches can be precisely controlled. In practice, the number N of neurons will be somewhere between n and n^2 for ε close to 1.

It is instructive to arrange the neurons in a rectangular matrix of n_x rows and n_y columns. Due to the distance cut, not all entries in the matrix actually have to contain a neuron. In any case it is desired that in the final stable state of the network at most one neuron is active in each column and in each row. If the efficiency of the channels is less than 1, it is of course possible that no partner can be found for a observation. In this case there might be rows or columns with no active neuron in the final state. If we declare neurons which occupy the same row or the same column as incompatible, it becomes obvious that solving the matching problem requires finding the maximal independent (compatible) subset of neurons which minimizes the sum of the associated distances. We set up the weights such that incompatible neurons inhibit each other; in order to save on the number weights, there are no activating links. Activation of the neurons is achieved by adjusting the thresholds or biases such that small distances yield large activation, and vice versa. A lower limit on the absolute value of the thresholds ensures activation of neurons with large distances if this is permitted by the compatibility structure of the network. For the sake of simplicity, the neurons are indexed by a single index.

$$\begin{aligned} w_{kl} &= \begin{cases} -1, & \text{if neuron } k \text{ and neuron } l \text{ are incompatible,} \\ 0, & \text{if neuron } k \text{ and neuron } l \text{ are compatible,} \end{cases} \\ w_{k0} &= (d_{\max} - \omega d_k)/d_{\max}, \quad 0 \leq \omega \leq 1. \end{aligned}$$

This choice of weights and thresholds guarantees that in the low temperature limit the potential activation of a neuron in the final state does not exceed its potential inhibition. Therefore two incompatible neurons cannot both settle in the active state. The lower limit of the negative threshold w_{k0} is equal to $1 - \omega$, attained if $d_k = d_{\max}$. We have chosen $\omega = 0.8$.

The final state of the network is a local minimum of the following energy function [4]:

$$E = - \sum_k \sum_l w_{kl} s_k s_l - \sum_k w_{k0} s_k,$$

where s_k denotes the state of neuron k . It follows from the form of the energy function that it is minimized when as many compatible neurons as possible are active and when the sum of the distances associated to the active neurons is minimal. In order to facilitate reaching the global minimum of the energy function thermal noise is introduced into the system. The standard mean-field approximation [8] leads to the well-known activation function

$$g(x) = \frac{1}{2} (1 + \tanh(x/T)),$$

with temperature T . Mean-field annealing requires a gradual cooling of the network. In order to speed-up the operation of the network the cooling schedule has to be carefully chosen. A simulation study has shown that the critical temperature T_c of the network is situated between $T = 0.5$ and $T = 1.0$, depending mainly on the overlap of the broken

Distribution of t	n	$R = 0.1$	$R = 0.25$	$R = 0.5$	$R = 1.0$
Normal	20	0.906	0.786	0.635	0.448
	50	0.898	0.779	0.626	0.438
	100	0.898	0.775	0.622	0.434
Uniform	20	0.905	0.782	0.626	0.422
	50	0.897	0.771	0.608	0.404
	100	0.896	0.765	0.600	0.395

Table 1: Fraction of correct matches as a function of R (MLP algorithm)

random sample (see below). We therefore start at a temperature somewhat above T_c , at $\log_{10} T_0 = 0.3$ or $T_0 \approx 2$. The network is cooled after each round of updates according to the formula $\log_{10} T_{k+1} = \log_{10} T_k - 0.2$ or $T_{k+1} \approx 0.63 \cdot T_k$. The final temperature is set to $\log_{10} T_f = -1.7$ or $T_f \approx 0.02$. Thus the final state is attained after 11 rounds of updates. The initial states of the neurons are random variates drawn from a uniform distribution in the interval $[0, 0.1]$. It has to be noted that synchronous or parallel updating may cause the network to cycle between two states. In order to avoid a reset algorithm we use asynchronous updating. In the final state, a neuron is considered to be active if its output exceeds a threshold of 0.75.

3 Performance of the network algorithm

The performance of the matching algorithm may be expected to depend mainly on the ratio R of the dispersion of x and y around t and the average distance D of adjacent values of t . We estimate D by the following formula which is exact for uniform t :

$$D = \frac{\tau\sqrt{12}}{n+1}.$$

For other types of distributions, e.g. the normal one, D is but a rough approximation. The ratio R , which we call the overlap, can now be defined as

$$R = \frac{\sqrt{\sigma_x^2 + \sigma_y^2}}{D}.$$

Small values of R indicate that the observations are on the average well separated if they are measured on the scale of the observation or measurement error.

Remarkably, the performance of the MLP algorithm for a given value of R depends indeed very little on the sample size and on the form of the distribution of t . Table 1 shows results of 5000 broken random samples for four values of R , three different sample sizes, and two different distributions of t (normal and uniform). In addition, we have chosen $\sigma_x = \sigma_y$. The entries in the table are fractions of correctly matched pairs.

We now proceed to compare the NN with MLP and SLSM. It should be stressed that in the case of a complete broken random sample the NN is *not* competitive with

		MLP		Neural Network			SLSM	
n	R	Matches	t	Matches	$\log_n N$	t	Matches	t
20	0.10	0.905	0.3	0.901	1.15	1.7	0.868	0.6
	0.25	0.782	0.3	0.774	1.29	2.6	0.716	0.6
	0.50	0.626	0.3	0.615	1.45	4.1	0.540	0.7
	1.00	0.422	0.3	0.416	1.62	8.0	0.345	0.9
50	0.10	0.897	0.4	0.893	1.12	5.0	0.862	1.7
	0.25	0.771	0.4	0.761	1.24	8.2	0.702	1.9
	0.50	0.608	0.4	0.595	1.36	14.6	0.520	2.0
	1.00	0.404	0.4	0.389	1.50	31.0	0.327	2.7
100	0.10	0.896	0.7	0.891	1.11	14.6	0.857	5.8
	0.25	0.765	0.7	0.759	1.21	23.1	0.695	6.2
	0.50	0.600	0.7	0.589	1.32	40.0	0.512	6.7
	1.00	0.395	0.7	0.381	1.43	91.1	0.321	8.0

Table 2: Comparison of MLP, SLSM, and Neural Network (complete samples)

either MLP or SLSM in terms of computing time. The MLP algorithm implies sorting of the components, requiring processing time of the order of $n \log n$. The processing time is independent of the overlap R of the sample. The SLSM algorithm implies sorting of all pairwise distances, in the worst case n^2 of them. The NN in the worst case has n^2 neurons and $(n-1)n^2$ weights, requiring a processing time of the order of n^3 , although for small values of R the number of neurons is much smaller than n^2 (see below). Table 2 shows the fraction of correctly matched pairs in 5000 random samples, for MLP, SLSM, and for the NN. The network is slightly worse than MLP, but not by more than a percentage point in nearly all cases. Also shown is the average of $\log_n N$, where n is the sample size and N is the number of neurons passing the distance cut, and the average CPU time in milliseconds per sample, measured on a DECstation 3000 AXP. The computing time of the NN scales approximately as $n^{1.5}$. The SLSM algorithm does considerably worse than the NN; it is, however, faster by a factor between 3 and 10.

We conclude that the NN gets very close to the statistical optimum in the case of complete samples. This gives us confidence that the NN performs equally well in cases where MLP cannot be applied, for instance if the broken random sample is incomplete or noisy. In this case, the SLSM algorithm is an alternative. Whether the price to be paid in computing time justifies the better performance of the network, depends of course on the application and on the computing power available.

First, we have investigated the performance of the NN as a function of the observation efficiency ε , down to a value of $\varepsilon = 0.7$, for three sample sizes and two values of the overlap. The distribution of t was chosen to be uniform, and the distance cut was set at the 99.9% quantile of the χ^2 -distribution with one degree of freedom. Note that the value of R is the nominal one; the effective value decreases with ε because of missing observations. Figure 1 shows the fraction of correct matches, normalized to the number n_2 of pairs actually present in each sample, for both the NN and the SLSM algorithm and

sample size $n = 50$. The results for $n = 20$ and $n = 100$ look very much the same and are not shown.

With the NN, there is very little effect for small values of the overlap R ($R=0.25$, figure 1a). The average fraction of correct matches drops by approximately 2% if ε is lowered to about 0.8, and then stabilizes. For larger values of R ($R=1.0$, figure 1a) the drop in the average fraction of correct matches is somewhat more pronounced. This can be explained by the fact that observations which have no partner can now interfere more easily with the matching of the paired observations, and sometimes may “snatch” one of the partners of a pair. Actually, we might as well consider single observations as noise with respect to the paired observations. In this interpretation lower values of ε correspond to larger relative amounts of noise, leading to a worse matching performance. However, this effect is compensated by a rise in matching probability which is due to R effectively decreasing with ε . Therefore only a slight dip around $\varepsilon \approx 0.85$ is observed. In any case, the loss in matching probability does not exceed 5 percentage points. The SLSM algorithm, starting at a lower value for $\varepsilon = 1.0$ actually gets better with decreasing efficiency. It catches up with the NN at $\varepsilon \approx 0.7$ for $R = 0.25$, and at $\varepsilon \approx 0.8$ for $R = 1.0$.

In all cases, the average network size parameter $\log_n N$ decreases nearly linearly with the observation efficiency (figure 1b). The processing time drops to about 50% if we pass from $\varepsilon = 1$ to $\varepsilon = 0.7$.

Finally, we have investigated the effect of noise on the performance of the network. To this purpose, each random sample of a total of 5000 was contaminated with additional observations according to the distribution of t . The number of noise observations in either component was drawn from a binomial distribution $\mathcal{B}_{n,\nu}$ with mean $n\nu$. The noise level ν was allowed to vary from 0 up to 0.3. Figure 2 shows the performance of the NN and of the SLSM algorithm as a function of ν , again for sample size $n = 50$. The fraction of correct matches is normalized to the nominal sample size n , which in this case is equal to the number of paired observations, since ε was chosen equal to 1. The value of R is the nominal one, corresponding to $\nu = 0$.

Even for small values of the overlap ($R=0.25$, figure 2a), the effect of the noise on the NN is a clear loss of matching probability, of the order of 20 percent. If the overlap is large ($R=1.0$, figure 2a), the fraction of correct matches drops by nearly 40 percent as ν approaches the upper limit of 0.3. Not shown in the figure is a strong rise of wrong matches, caused by accidental matches of noise observations with either noise or partners of a pair. The average network size parameter rises about linearly with the noise level (figure 2b).

The performance of the SLSM algorithm also gets worse with increasing noise level, albeit at a lower rate than with the NN. If $R = 0.25$, it catches up with the NN at $\nu \approx 0.3$; if $R = 1.0$, it is as good as the NN above $\nu \approx 0.2$.

4 Summary

We have constructed a recurrent neural network which matches bivariate broken random samples. The method may be generalized to paired multidimensional observations in a straightforward way.

For complete samples the performance of the network nearly attains the statistical optimum, given by maximum likelihood pairing. In practice, however, it is to be expected that the sample is incomplete or noisy. We have therefore investigated the performance for different levels of observation efficiency and noise. The effect of missing observations is a slight loss of matching probability, of the order of a few percentage points, stabilizing at an observation efficiency of about 0.85. It is nearly independent of sample size and of the overlap R . In contrast, the effect of noise does depend on the overlap R , although not much on the sample size. For larger values of R the performance deteriorates appreciably with increasing amounts of noise.

We also have investigated a suboptimal but simple matching algorithm, called “Sequential Least-Squares Matching” (SLSM), which can be applied to incomplete or noisy samples. The network performs better than SLSM for observation inefficiencies above 70 percent and noise levels below 25 percent, although the improvement is not spectacular. The price to be paid for the better performance is a rise in computing time by a factor varying between 3 and 10 for samples sizes up to 100.

References

- [1] M.H. DeGroot, P.I. Feder and P.K. Goel, *Ann. Math. Statist.* 42 (1971) 578.
- [2] M.C. Chew, *Ann. Statist.* 1 (1973) 433.
- [3] P.K. Goel, *Ann. Statist.* 3 (1975) 1364.
- [4] J.J. Hopfield and D.W. Tank, *Biolog. Cybern.* 52 (1985) 1.
- [5] Y. Shrivastava, S. Dasgupta and S.M. Reddy, *IEEE Trans. Neural Networks* 3 (1992) 951.
- [6] C. Peterson and B. Söderberg, *International Journal of Neural Networks* 1 (1989) 3.
- [7] J.J. Hopfield, *Proc. Natl. Acad. Sci. USA* 79 (1982) 2554.
- [8] C. Peterson, *Connection Science* 3 (1991) 3.
- [9] R. Frühwirth, *Comp. Phys. Comm.* 78 (1993) 23.

Figure 1: Network performance as a function of the observation efficiency ϵ . n is the sample size, R is the overlap of the random sample.

Figure 2: Network performance as a function of the noise level ν . n is the sample size, R is the overlap of the random sample.