Investigating the Use of Nearest-Neighbor Interpolation for Cancer Research

Matthias Fuchs
Center for Learning Systems
and Applications (LSA)
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049, 67653 Kaiserslautern
Germany
`fuchs@informatik.uni-kl.de`

Stefan Forster
Institut für Humangenetik
Universität des Saarlandes
66421 Homburg
Germany
`hgsfor@med-rz.uni-sb.de`

March 6, 1997

Abstract

We investigate in how far interpolation mechanisms based on the nearest-neighbor rule (NNR) can support cancer research. The main objective is to use the NNR to predict the likelihood of tumorigenesis based on given risk factors. By using a genetic algorithm to optimize the parameters of the nearest-neighbor prediction, the performance of this interpolation method can be improved substantially. Furthermore, it is possible to detect risk factors which are hardly or not relevant to tumorigenesis. Our preliminary studies demonstrate that NNR-based interpolation is a simple tool that nevertheless has enough potential to be seriously considered for cancer research or related research.

1 Introduction

In spite of tremendous progress in evaluating the underlying parameters of bioscientific mechanisms still a lot of work remains to be done to understand the complexity of the functional background. In this context, one of the biggest challenges in biological research is to find the key for tumorigenesis. Thousands of facts for the development of different tumors could be collected over the past century. But still nobody can explain the underlying complexity of mechanisms and tumor-depending functional diversities of carcinogenesis in detail.

One tool to shed more light on these hidden functions of life could be the use and improvement of methods originating from artificial intelligence and computational learning strategies. As mentioned above, cancer development is still a major problem in medical science. While a lot of different environmental factors are known to be definitely causative for certain tumors, many other unknown biomedical parameters must also be involved. Even if all the parameters participating in tumor development were known, the scientists would have problems identifying the functional connections between them. It is furthermore very difficult to give a prediction for the probability of a development of a certain kind of tumor in a certain patient even when all his so-called *risk factors* are known.

Keeping all these immense problems in mind we tried to find out in how far intelligent computer programs can be able to support scientific research in this area. Three main questions arise in this context:

1. Can computer programs give us reliable data concerning the probability of cancer development in a certain person if risk factors of the patient are known?
2. Can these programs help us find new risk factors in a given pool of suspected or at first sight harmless factors (parameters)?
3. Can such programs give us some help in future days to identify underlying functional connections between responsible factors of tumorigenesis?

As for question 1, we are essentially interested in a computer program (or a function) that computes the probability of tumor development when given the risk factors of a patient. Commonly, n risk factors are represented by n real values $v_1, \ldots, v_n \in \mathbb{R}$ or by a vector $(v_1, \ldots, v_n) = \vec{v} \in \mathbb{R}^n$. The function P computing the probability for tumor development can be specified by $P : \mathbb{R}^n \to [0; 1]$.

Usually, such a function P is derived or *learned* from a given set of m input/output samples $S = \{(\vec{v}_i, p_i) \in \mathbb{R}^n \times [0; 1] \mid 1 \leq i \leq m\}$. It is obvious that the success of learning P depends on the quality and quantity of the data in S. Furthermore, for medical purposes it can be assumed that S will be rather large and will be continuously updated and in particular extended. Therefore, it is advisable to use methods that are quite tolerant regarding modifications of S and have efficient incremental learning capabilities.

Methods based on the so-called *nearest-neighbor rule* (NNR, [6]) satisfy these conditions. We shall explain in section 2 how the NNR can be utilized to design P. At that point we shall have addressed question 1 from above. Section 3 will demonstrate that the NNR approach also allows us to deal with question 2. (Question 3, however, can hardly be addressed with a NNR approach.) We illustrate the approach with a case study in human bladder cancer. Sections 4 and 5 explain the experimental set-up and report on the experimental results, respectively. A discussion in section 6 concludes this report.

2 Basics of the NNR Approach

Methods based on the NNR fascinate with their simplicity. Nonetheless (or maybe because of that) they have been applied in many variations with considerable success. The NNR is probably best known for its application to classification tasks (e.g., [1, 20]), but has also been successfully used for predicting (i.e., interpolating or estimating) real-valued attributes (e.g., [12, 16, 4]). The latter kind of application has been studied thoroughly by a number of mathematicians (e.g., [7, 9, 19, 5]) so that there is not much to add from a theoretician's point of view. Theoretical examinations, however, can only reveal general tendencies and give general results (e.g., average/best case or asymptotic behavior). Usefulness for a specific application must be investigated experimentally. The objective of this report is to examine NNR-based interpolation as to its applicability in cancer research.

Section 1 already outlined the scenario in which the NNR is to be deployed: An unknown function $\Psi : \mathbb{R}^n \rightarrow [0;1]$ is characterized by a finite sample S of its input/output behavior, i.e., $S = \{(\vec{x}_i, y_i) \in \mathbb{R}^n \times [0;1] \mid 1 \leq i \leq m\}$, where $\Psi(\vec{x}_i) = y_i$. In our specific case $\Psi(\vec{x}) = y$ is the probability for a person to develop a certain type of cancer (within a specified period of time) in the presence of his or her n risk factors represented by $\vec{x}$. A function P based on the NNR is to approximate Ψ as closely as possible using the sample set S. Obviously, when given $\vec{x} = \vec{x}_i$ for some $1 \leq i \leq m$, $P(\vec{x}) = y_i$ is the best choice. Otherwise, when given $\vec{x} \notin \mathcal{I} = \{\vec{x}_1, \ldots, \vec{x}_m\}$, a slight modification of the so-called k-NNR is employed: Let $\mathcal{N} = \{\vec{z}_1, \ldots, \vec{z}_q\} \subseteq \mathcal{I}$, $\delta : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ a distance measure (commonly Euclidean distance which we also use here), and $q \geq k \geq 1$ so that $\delta(\vec{x}, \vec{z}_i) \leq \delta(\vec{x}, \vec{z}_{i+1})$, $\delta(\vec{x}, \vec{z}) > \delta(\vec{x}, \vec{z}_k)$ for all $\vec{z} \in \mathcal{I} \setminus \mathcal{N}$, and $\delta(\vec{x}, \vec{z}_k) = \delta(\vec{x}, \vec{z}_j)$ for all $k \leq j \leq q$. In other words, $\mathcal{N}$ is the set of k nearest neighbors of $\vec{x}$ (w.r.t. S), extended by all those neighbors with a distance from $\vec{x}$ that is equal to that of the kth nearest neighbor. Thus we avoid the common practice of "arbitrarily" breaking ties between all those $\vec{z} \in \mathcal{I}$ that qualify as the kth nearest neighbor of $\vec{x}$. (In our opinion it is not justifiable to select—arbitrarily or otherwise—just one from several possible kth nearest neighbors, thus preventing potential kth nearest neighbors from contributing to the subsequent computation of $P(\vec{x})$.)

The value $P(\vec{x}) = y$ is computed as usual as a weighted average of the nearest neighbors.

More precisely, let $\hat{y}_i = \Psi(\vec{z}_i)$ (according to S). Then we have

$$P(\vec{x}) = \left(\sum_{i=1}^{q} w_i \cdot \hat{y}_i\right) / \left(\sum_{i=1}^{q} w_i\right)$$

Commonly, $w_i \geq w_{i+1}$ so that nearer neighbors have a bigger effect. We chose weights that center on the maximal distance $d_{max} = \delta(\vec{x}, \vec{z}_k)$ of the nearest neighbors:

$$w_i = d_{max} + 1 - \delta(\vec{x}, \vec{z}_i)$$

This appears to be more sensible than using preset constant weights because it allows for taking into account absolute differences of distances as opposed to the rather coarse information reflected by ordinals.

NNR-based interpolation has the obvious advantage that it can very easily cope with modifications and extensions of S. Modifying an element of S or adding further elements to S does not require any particular action. This property originates from the explicit representation of the function to be learned, namely by the sample set S. Other approaches that use S more implicitly by encoding it in some structure (during a learning phase, e.g., connections and weights of a neural network, or (LISP) programs in genetic programming [17]) basically have to start from scratch after each modification or extension.

The downside of an explicit use of S is a large storage requirement and query times that increase linearly with the size of S (and with the number n of risk factors). With appropriate implementation techniques (e.g., [13]), however, these problems (in particular query time) can be handled quite satisfactorily.

3 Optimizing Nearest-Neighbor Interpolation

The approach presented in the preceding section allows us to predict results of $\Psi(\vec{x})$ for $\vec{x}$ not present in the sample set S with the help of P. Thus, question 1 posed in section 1 can be coped with. Nevertheless, the predictions of P are based on an implicit simplifying assumption: *all risk factors are equally important*. This assumption is reflected by the fact that the (Euclidean) distance measure δ which is pivotal for computing $P(\vec{x})$ treats all risk factors uniformly since

$$\delta(\vec{x}, \vec{z}) = \sqrt{\sum_{i=1}^{n}(x_i - z_i)^2}$$

However, applications of NNR-based approaches in the area of classification have taught us that not all attributes (risk factors in our case) are equally relevant to prediction. This means that some risk factors are more important than others and therefore should influence $P(\vec{x})$ more. As a matter of fact, it may be (and often is) the case that some alleged risk factor actually is not a risk factor at all. Nonetheless it influences the computation of distance and consequently affects $P(\vec{x})$. Less important or completely irrelevant risk factors can have severely negative effects on prediction accuracy.

The most obvious way to tackle the problem is to modify the source of it, namely the distance measure δ (e.g., [15, 22]). For this purpose we generalize δ as follows:

$$\delta(\vec{x}, \vec{z}) = \sqrt{\sum_{i=1}^{n} c_i \cdot (x_i - z_i)^2}\,, \quad c_i \geq 0$$

Instead of the "default" values $c_i = 1$, the coefficients c_i can be chosen so as to reflect the importance of the respective risk factor i. The bigger c_i is, the more risk factor i affects δ and consequently P. With $c_i = 0$, risk factor i can be completely ignored because it does not affect δ anymore. Note that we have the special case "factor selection" if we restrict the c_i to values from $\{0, 1\}$ (cp. [23]).

Unfortunately, the importance of risk factors is not known a priori and is actually a prominent problem especially in cancer research (cp. question 2 in section 1). These considerations lead us back to the original goal of modifying the distance measure, namely to increase prediction accuracy. If we can find coefficients $c_1, \ldots, c_n$ so that a (nearly) optimal prediction accuracy is attained, it is reasonable to assume that these coefficients then reflect the importance of the respective risk factors. Therefore, we search for $c_1, \ldots, c_n$ so that P offers optimal prediction accuracy.

To this end we employ a *genetic algorithm* (GA; [14, 8]). The use of a GA appears to be appropriate because the GA has the potential to cope with intricate search spaces in the absence of any knowledge about their structure. Furthermore, a GA is less prone to getting trapped in a local optimum. Both properties are highly valuable for our purpose. In the sequel, we describe the basics of the GA and its application to our optimization problem.

Unlike other search methods, the GA maintains a set of (sub-optimal) solutions, i.e., several points in the search space. In this context, a solution is preferably called an *individual*, and the whole set is referred to as a *population* or *generation*. Usually, the size of the population is fixed. In order to explore the search space, the GA applies so-called *genetic operators* to (a subset of the) individuals of its current population. Thus new individuals can be created and hence new points in the search space can be reached. In order to keep the population size fixed, it must be determined which individuals are to be eliminated in order to make room for the new ones. For this purpose a so-called *fitness measure* is employed which rates the fitness (i.e., the ability to solve the problem at hand) of each individual of the current population. The genetic operators are applied to the fittest individuals, this way producing offspring which then replaces the least fit individuals ("*survival of the fittest*").

So, the GA basically proceeds as follows: Starting with a randomly generated initial population, the GA repeats the cycle comprising the rating of all individuals using the fitness measure, applying the genetic operators to the best individuals, and replacing the worst individuals with offspring of the best, until some termination condition is satisfied. The pool of best or "surviving" individuals is determined by the survival rate r.

The percentage r of best individuals is simply copied from the current population to the successor population. From this pool, the genetic operators draw required parent individuals at random (*elite* or *truncate selection*) in order to produce new individuals to restock the population.

Here, we are searching for suitable coefficients $c_1, \ldots, c_n$ to optimize the accuracy of P. Besides these coefficients, also the number k of nearest neighbors to be considered affects P. k is in general difficult to determine. Therefore, k is also subject to search. We occasionally write c_0 instead of k to ease notation. Hence, an individual Υ is represented by $\Upsilon = (c_0, \ldots, c_n)$.

Three genetic operators are used: *crossover*, *mutation*, and *hill-climbing*. **Crossover** and **mutation** are the "standard" operators of a GA. Crossover produces a new individual $\Upsilon = (c_0, \ldots, c_n)$ from two distinct parent individuals $\Upsilon_1 = (c_0^{(1)}, \ldots, c_n^{(1)})$ and $\Upsilon_2 = (c_0^{(2)}, \ldots, c_n^{(2)})$ by applying "multiple-point" crossover: $c_i = c_i^{(1)}$ or $c_i = c_i^{(2)}$ with a probability of 50%, respectively. The resulting individual Υ may be subject to mutation with a probability P_{mut}. If Υ is actually subject to mutation, one of the c_i is selected at random and replaced with a value also generated at random. Naturally, certain range restrictions apply, i.e., $c_i^{min} \leq c_i \leq c_i^{max}$.

We also use **hill-climbing** (in form of a genetic operator) to profit from potential benefits of hill-climbing. Hill-climbing produces a new individual Υ from a parent individual $\hat{\Upsilon} = (\hat{c}_0, \ldots, \hat{c}_n)$ by selecting a $\hat{c}_i$ at random and incrementing or decrementing $\hat{c}_i$, i.e., $\Upsilon = (\hat{c}_0, \ldots, \hat{c}_{i-1}, \hat{c}_i \pm 1, \hat{c}_{i+1}, \ldots, \hat{c}_n)$. ± 1 denotes $+1$ or -1 with a probability of 50%, respectively. If range restrictions are violated, then we deterministically use $+1$ or -1 as the case may be. (Mutation is never used in connection with hill-climbing.) Crossover or hill-climbing are employed with a probability of P_{co} and $P_{hc} = 100\% - P_{co}$, respectively, when producing a new individual Υ.

The **fitness** of an individual $\Upsilon = (c_0, \ldots, c_n)$ is measured in terms of the prediction accuracy of P when using $k = c_0$ and the coefficients $c_1, \ldots, c_n$ for the distance measure δ. Naturally, we can only center on the information provided by S. We employ the so-called *leave-one-out* method: For each $(\vec{x}_i, y_i) \in S$ compute $P(\vec{x}_i)$ when using $S_i := S \setminus \{(\vec{x}_i, y_i)\}$. We write P_{S_i} in order to make clear that P is based on S_i instead of the whole set S. $|P_{S_i}(\vec{x}_i) - y_i|$ is the *prediction error*. With $m = |S|$,

$$\varepsilon_{avg}(\Upsilon) = \frac{1}{m} \sum_{i=1}^{m} |P_{S_i}(\vec{x}_i) - y_i| \quad \text{and} \quad \varepsilon_{max}(\Upsilon) = \max(\{|P_{S_i}(\vec{x}_i) - y_i| \mid 1 \leq i \leq m\})$$

are the *average* and *maximal* prediction error of individual Υ, respectively. An individual Υ is considered fitter than an individual $\hat{\Upsilon}$ if

$$\varepsilon_{avg}(\Upsilon) < \varepsilon_{avg}(\hat{\Upsilon}) \qquad \text{or} \qquad \varepsilon_{avg}(\Upsilon) = \varepsilon_{avg}(\hat{\Upsilon}) \quad \text{and} \quad \varepsilon_{max}(\Upsilon) < \varepsilon_{max}(\hat{\Upsilon})\,.$$

Using this fitness measure, the GA searches for a k and coefficients $c_1, \ldots, c_n$ that minimize the average and maximal prediction error (primary and secondary objective, respectively) and thus optimize prediction accuracy of P with respect to S and the leave-one-out method. It is reasonable to assume that such an optimization can also increase overall accuracy when being given an arbitrary $\vec{x}$. In the sequel we present our experimental studies which support this assumption.

4 Experimental Set-up

In order to test our approach for approximating Ψ (i.e., the probability to be affected with cancer) based on a finite sample S of the input/output behavior of Ψ, we of course have to create an "artificial" Ψ_A. In reality, we can only expect to create a finite sample with the help of polls and statistical evaluations. This is all we need to approximate a "real" Ψ, but it is insufficient to test and illustrate our approach. Obviously, an artificial Ψ_A should reflect essential properties present in reality, i.e., central mathematical properties which affect prediction accuracy (e.g., slope) should not differ significantly.

For this reason we selected bladder cancer for our experimental studies. Bladder cancer has been studied thoroughly (e.g., [18, 21]) and therefore offers reliable epidemiological and statistical data with which we can expect to create an artificial, but realistic Ψ_A. The six risk factors considered here are *sex*, *age*, *smoking behavior*, *occupational risks* (e.g., exposition to toxic substances), *alcohol consumption*, and *predisposition*. Each risk factor can take on one out of a finite number of discrete values. (Discrete values for risk factors are very common. It is unrealistic to assume that medical examinations or polls will provide us with continuous values. Besides, certain factors, like sex, are inherently discrete.) Where necessary, discrete values are obtained by grouping continuous values appropriately.

Studies in bladder cancer revealed that there is a basic probability p_B (the so-called *incidence*) to be affected with bladder cancer. p_B ranges between $3 \cdot 10^{-5}$ and $6 \cdot 10^{-5}$ for a 20 year-old woman with no further risk factors in Western Europe and North America (cf. [18]). According to the studies (e.g., [18, 21]), each value of a risk factor causes a multiplicative increase of p_B. For instance, being a man (at least) doubles the risk of bladder cancer. Therefore, we create an artificial Ψ_A as follows:

Let $D_1, \ldots, D_n$ be the discrete values for risk factors $1, \ldots, n$, respectively (here $n = 6$). Let $f_i(j) \in \mathbb{R}$ be the factor by which p_B is to be multiplied if risk factor i takes on the value $j \in D_i$. Each $f_i(j)$ is based on epidemiological data. Then for each $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{D} = D_1 \times \cdots \times D_n$ we set

$$\Psi_A(\vec{x}) = p_B \cdot \prod_{i=1}^{n} f_i(x_i) \in [0;1] .$$

Thus, we obtain a "complete sample" $\mathcal{C} = \{(\vec{x}, y) \mid \vec{x} \in \mathcal{D},\, y = \Psi_A(\vec{x})\}$ of the input/output behavior of an artificial Ψ_A. From this complete sample $\mathcal{C}$ we can extract partial samples $S \subset \mathcal{C}$ and can check how well P approximates Ψ_A depending on the size of S and on optimizations described in section 3.

Table 1: Average and maximal prediction error (top and bottom, respectively) for default coefficients and optimized coefficients depending on the size of a sample in %.

size%	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	opt.
20%	0.015 0.193	0.014 0.204	0.013 0.208	0.014 0.212	0.015 0.214	0.015 0.219	0.015 0.217	0.015 0.223	0.015 0.222	0.009 0.132
40%	0.007 0.137	0.007 0.141	0.007 0.159	0.008 0.170	0.008 0.175	0.009 0.190	0.009 0.191	0.009 0.198	0.010 0.199	0.003 0.077
60%	0.004 0.103	0.003 0.109	0.004 0.115	0.004 0.125	0.004 0.143	0.004 0.149	0.005 0.154	0.005 0.160	0.005 0.172	0.001 0.057
80%	0.002 0.064	0.001 0.078	0.002 0.085	0.002 0.087	0.002 0.092	0.002 0.101	0.002 0.110	0.002 0.113	0.002 0.118	0.000 0.028

5 Experimental Results

We conducted experiments so as to illustrate the approach and to examine the quality of the approximation P of a Ψ_A in dependence of the sample $S \subset \mathcal{C}$ and the parameter k (number of nearest neighbors to be considered), as well as the coefficients $c_1, \ldots, c_n$ of the distance measure δ. For the experiments we chose $|\mathcal{D}| = |\mathcal{C}| = 504$. A rather small size is necessary to obtain experimental results in an acceptable period of time. Although $P(\vec{x})$ can be computed quite fast even for larger sizes of S (e.g., in 0.1 seconds for $|S| = 50,000$ on a SPARCstation 20 with a sub-optimal implementation of the NNR), $P(\vec{x})$ must be computed for all $\vec{x} \in \mathcal{C}$ so as to determine a mean prediction error. In order to check on influences of the choice of S, this is done for t sets $S \subset \mathcal{C}$ so that $P(.)$ is computed $t \cdot |\mathcal{C}|$ times.

First, we used default values $c_1 = \cdots = c_n = 1$ and had k range from 1 to 9. We normalized the risk factors to avoid a possible bias regarding distance measurement on account of range differences between the D_i. More precisely, we normalized each D_i to $[0;1]$ when using P, i.e., each $\vec{x} \in \mathcal{D}$ was normalized to $\tau(\vec{x}) \in [0;1]^n$. As usual, $|P(\tau(\vec{x})) - \Psi_A(\vec{x})|$ is the (absolute) prediction error.

We selected $S \subset \mathcal{C}$ so that the percentage of samples in S was 20%, 40%, 60%, or 80%. For each of these sizes, 100 sample sets $S \subset \mathcal{C}$ were chosen at random. With each such sample set S, $P(\tau(\vec{x}))$ was computed for all $\vec{x} \in \mathcal{D}$. Thus, the average and maximal prediction error for P (with respect to the 100 sample sets) could be determined. This information is given in table 1. Each entry shows the average prediction error (top) and the maximal prediction error (bottom). The heads of columns 2–10 show the respective value of parameter k. The head of each row lists the sample percentage $100 \cdot |S|/|\mathcal{C}|$. As expected, average and maximal prediction error drop notably with an increasing sample percentage. Furthermore, the best results are achieved with rather small values for k. Increasing k slowly, but continuously decreases prediction accuracy.

When optimizing the coefficients $c_1, \ldots, c_n$ and parameter k as described in section 3, substantial improvements can be achieved.[1] Although we used a sample set S which merely comprised 20% of $\mathcal{C}$, the optimized parameters allowed for significantly reducing both average and maximal prediction error also when used for larger sample sets. The last column of table 1 shows the results obtained with optimized parameters. It is worth noting that the "optimal" $k = 1$.

Apart from the mentioned improvements of prediction accuracy, the optimized coefficients also allowed us to draw conclusions regarding the importance of risk factors. The most prominent result was that the coefficient for risk factor 'alcohol consumption' was 0 which indicates that alcohol consumption does not (notably) influence bladder cancer. This observation—although already known in connection with bladder cancer—supports our belief that the optimized NNR approach also can help us to identify more and less important risk factors, thus helping us to answer question 2 in section 1 which essentially addresses a data-mining problem.

6 Discussion

Interpolation mechanisms based on the k nearest-neighbor rule (k-NNR) have proven their usefulness in a variety of applications (e.g., [10, 4, 11]). In medical research, statistical methods like multiple (linear) regression appear to be dominant in order to detect and describe functional dependencies between input and output variables (e.g., [3]). These methods, however, assume a certain kind of functional dependency (usually a linear polynomial in the input variables) and then adapt available parameters appropriately (see also [2]). Methods based on the k-NNR are not restricted to a certain a priori chosen type of functional dependency. They essentially can approximate arbitrary (continuous) functions (cp. [7, 9, 5, 19]).

Most interpolation methods (e.g., regression, genetic programming, neural networks) generate an implicit representation of the given sample of the input/output behavior of the (unknown) function which is to be approximated (e.g., coefficients of a polynomial, a program, weights and topology of a neural net). Methods based on the k-NNR, however, use the sample in an explicit manner. The advantage is that there is no need for pre-processing. Therefore, on the one hand k-NNR methods are very well-suited to frequently changing samples. On the other hand, k-NNR methods have rather high storage requirements and usually are not as efficient as pre-processing methods which create a more compact and also more efficient approximator during the very pre-processing phase. With suitable implementation techniques (e.g., [13]), however, k-NNR interpolation can gain an acceptable level of efficiency.

Our preliminary studies with the k-NNR in the area of cancer research demonstrated that k-NNR methods are useful for predicting the likelihood of tumorigenesis based on given risk factors. By optimizing the "standard" k-NNR using a genetic algorithm,

[1] Parameter settings: population size 50, 10 generations, $r = 30\%$, $P_{co} = P_{hc} = 50\%$, $P_{mut} = 10\%$, $c_0^{min} = 1$, $c_0^{max} = 10$, (recall $c_0 = k$), $c_i^{min} = 0$, $c_i^{max} = 100$ for $1 \leq i \leq 6$.

prediction accuracy could be improved significantly. The optimization also allows for identifying more or less relevant risk factors. (This can of course also be achieved by statistical methods like multiple correlation and regression analysis.)

Thus, two out of three important questions in cancer research posed at the beginning can be addressed with a k-NNR method. The third question concerning the identification of functional connections between risk factors, however, cannot be answered with a k-NNR method. Genetic programming [17] could be helpful for dealing with this question in that *explicit* functions (programs) are computed, whereas neural networks and statistical methods (the latter mostly only permitting to test how well an *assumed* functional connection fits the given data) also appear to be useless in this context.

Finally, the main purpose of this report is to point out the potential of k-NNR methods for cancer research and related research. We do not claim that these methods are necessarily better or more suitable than any other method. But their simplicity and their advantages discussed earlier make them worthwhile considering. In this context, a thorough and fair comparison of interpolation methods would be helpful. Given the vast number of methods and possible comparison criteria, such a comparison amounts to a research project of its own.

References

[1] **Aha, D.W.; Kibler, D.; Albert, M.K.:** *Instance-Based Learning Algorithms*, Machine Learning **6:**37–66, 1991.

[2] **Andreassen, S.; Benn, J.; Hovorka, R.; Olesen, K.; Carson, E.:** *A Probabilistic Approach to Glucose Prediction and Insulin Dose Adjustment: Description of Metabolic Model and Pilot Evaluation Study*, Comp. Methods Programs Biomed. **41:**153–165, 1994.

[3] **Assmann, G.; Schulte, H.; von Eckardstein, A.:** *Hypertriglyceridemia and Elevated Lipoprotein(a) Are Risk Factors for Major Coronary Events in Middle-aged Men*, Am. J. of Cardiology **77:**1179–1184, 1996.

[4] **Atkeson, C.G.:** *Using Locally Weighted Regression for Robot Learning*, Proc. International Conf. on Robotics and Automation, IEEE Press, 1991, pp. 958–963.

[5] **Cheng, P.E.:** *Strong Consistency of Nearest Neighbor Regression Function Estimators*, Journal of Multivariate Analysis **15:**63–72, 1984.

[6] **Cover, T.M.; Hart, P.E.:** *Nearest Neighbor Pattern Classification*, IEEE Transactions on Information Theory, Vol. IT-13, Jan. 1967, pp. 21–27.

[7] **Cover, T.M.:** *Estimation by the Nearest Neighbor Rule*, IEEE Transactions in Information Theory, Vol. IT-14, No. 1, Jan. 1968, pp. 50–55.

[8] **De Jong, K.:** *Learning with Genetic Algorithms: An Overview*, Machine Learning **3:**121–138, 1988.

[9] **Devroye, L.P.:** *The Uniform Convergence of Nearest Neighbor Regression Function Estimators and Their Application in Optimization*, IEEE Transactions on Information Theory, Vol. IT-24, No. 2, March 1978, pp. 142–151.

[10] **Farmer, J.D.; Sidorowich, J.J.:** *Predicting Chaotic Time Series*, Physical Review Letters, Vol. 59, No. 8, August 1987, pp. 845–848.

[11] **Fechteler, T.; Dengler, U.; Schomburg, D.:** *Prediction of Protein Three-dimensional Structures in Insertion and Deletion Regions: A Procedure for Searching Data Bases of Representative Protein Fragments Using Geometric Scoring Criteria*, J. of Molecular Biology **253:**114–131, 1995.

[12] **Franke, R.:** *Scattered Data Interpolation: Tests of Some Methods*, Mathematics of Computation, Vol. 38, No. 157, January 1982.

[13] **Friedman, J.H.; Bentley, J.L.; Finkel, R.A.:** *An Algorithm for Finding Best Matches in Logarithmic Expected Time*, ACM Trans. on Mathematical Software 3(3), Sept. 1977, pp. 209–226.

[14] **Holland, J.H.:** *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, Ann Arbor: Univ. of Michigan Press, 2^{nd} edition, 1992.

[15] **Kelly, J.D.; Davis, L.:** *Hybridizing the Genetic Algorithm and the K Nearest Neighbors Classification Algorithm*, Proc. 4^{th} International Conference on Genetic Algorithms (ICGA-91), 1991, Morgan Kaufmann, pp. 377–383.

[16] **Kibler, D.; Aha, D.W.; Albert, M.K.:** *Instance-based Prediction of Real-valued Attributes*, Comput. Intell. **5**:51–57, 1989.

[17] **Koza, J.R.:** *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, 1992.

[18] **Kroft, S.H.; Oyasu, R.:** *Biology of Disease, Urinary Bladder Cancer: Mechanisms of Development and Progression*, Laboratory Investigation, Vol. 71, No. 2, 1994, pp. 158–170.

[19] **Li, K.-C.:** *Consistency for Cross-validated Nearest Neighbor Estimates in Nonparametric Regression*, The Annals of Statistics, 1984, Vol. 12, No. 1, pp. 230–240.

[20] **Michie, D.; Spiegelhalter, D.J.; Taylor, C.C.:** *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994.

[21] **Porru, S.; Aulenti, V.; Donato, F.; Boffetta, P.; Fazioli, R.; Cunico, S.C.; Alessio, L.:** *Bladder Cancer and Occupation: A Case-control Study in Northern Italy*, Occupational and Environmental Medicine **53:**6–10, 1996.

[22] **Raymer, M.L.; Punch, W.F.; Goodman, E.D.; Kuhn, L.A.:**, *Genetic Programming for Improved Data Mining – Application to the Biochemistry of Protein Interactions*, Proc. First International Conference on Genetic Programming (GP-96), MIT Press, 1996, pp. 375–380.

[23] **Siedlecki, W; Sklansky, J.:** *A Note on Genetic Algorithms for Large-scale Feature Selection*, Pattern Recognition Letters **10**:335–347, November 1989.