Institute of Phonetic Sciences,
University of Amsterdam,
Proceedings 24 (2000-2001), 125-146.

ON THE PROBABILISTIC ORDERING OF CONSTRAINTS

Louis ten Bosch

Abstract

In this paper, we will discuss a number of aspects related to the probabilistic orderings of constraints. The developed model is referred to as Probabilistic Ranking Optimisation (PRO). The treatment of ranking in Optimality Theory is taken as a starting point, but the emphasis here will be on the mathematical properties of ranking solutions and the connection with adaptive and sequential learning. While in Optimality Theory constraints are (linearly) ranked along a one-dimensional continuum, the current exposition is not constrained to a one-dimensional continuum, but can be applied in a more general setting. A relation between the learnability of constraints on the one hand and aspects of graph theory on the other is established. The resulting PRO model enables to understand the modelling power of ranking in terms of the number and structure of probability properties that have to be fulfilled.

1 Introduction

This paper deals with orderings, and more specifically, a probabilistic model for orderings. The model aims at a description of observed orderings of objects or events. As the first example, we consider the following morning activities (example 1):

‘a’ = ‘wake up’
‘b’ = ‘take a shower’
‘c’ = ‘eat bread’
‘d’ = ‘brush teeth’
‘e’ = ‘go to work’

Each of these activities is denoted by a symbol. The ordering of these activities on five working days can be represented by a list:

L = {‘abcde’, ‘acbde’, ‘adbce’, ‘adcbe’, ‘abcde’}

In this form, L just describes five working days, but L can also describe the activities over a longer time span, by e.g. allotting probabilities to each of its elements. In the latter probabilistic interpretation, we may conclude that action ‘a’ and ‘e’ are ‘likely’ to be the first and last action, respectively, while the ordering of remaining actions is less well specified and may actually differ from day to day.

The ordering of the activities along a time axis is *total*, which means that on each day, each activity 'x' is simultaneous with itself, any two activities 'x' and 'y' are always comparable in the sense that 'x' occurs earlier than 'y', simultaneous with 'y', or later than 'y', and there is transitivity: if 'x' is earlier than 'y' and 'y' is earlier than 'z', then 'x' is earlier than 'z'. In the example above, all activities on any day are comparable with respect to their (time) ordering.

The symbols 'a', 'b', … may also refer to *transformations*, acting on an object, rather than to activities. For example, when this object is a phonetic-phonological representation of a pronunciation, a possible interpretation of four symbols reads (example 2):

'a' = 'devoice final obstruent if voiced'
'b' = 'diphthongize vowel in final syllable'
'c' = 'reduce penultimate syllable'
'd' = 'apply vowel harmony in penultimate and final syllable'

The application of one or more actions on a certain representation will alter this representation into another one. This resulting representation usually depends on the *ordering* of the chosen actions. In general, many possibly different outcomes can be generated from one input representation, by choosing another ordering of these transformations. The following example shows the effect of the switch of two transformations on the third singular form of 'praten' ('talk') en 'raden' ('guess') in Dutch.

Degemination:	Delete one of two identical consonants.
Devoicing:	Devoice an obstruent (in the coda).

Underlying representation:	/pra:t+t/	/ra:d+t/
Devoicing	[pra:t+t]	[ra:t+t]
Degemination	[pra:t]	[ra:t]

The reverse ordering of these constraints produces a different result (which is incorrect, at least for Dutch):

Degemination	[pra:t]	[ra:d+t]
Devoicing	[pra:t]	[ra:t+t]

This shows that a correct ordering of the transformations is essential to obtain (explain, predict) the observed phonological surface forms.

If we assume that it is possible to figure out, given an output representation, the ordering in which transformations have been applied to the underlying representation to obtain the output representation, then we obtain a mapping from the output 'surface' representation to a sequence of symbols[1]. As a result, the list L 'encodes' the probabilistic structure in the stream of incoming representations in terms of the

[1] This is a substantial assumption. It can be related to the idea that each action can be modelled as a transducer that maps the input representation to a sequence of 'marks' indicating the presence or absence of a mismatch between the representation and the action. When certain restrictions are met, ranking models can be interpreted as regular relations, which would simplify this mapping (see Frank & Satta, 1998).

transformations. If we assume that for each input representation *all* transformations must be used in some order, the ordering is again total.

A slightly different interpretation of the symbols deals with *properties* or constraints rather than with actions, such as in (example 3):

'a' = 'the final obstruent *is* unvoiced'
'b' = 'in the final syllable the vowel *is* long'
'c' = 'the penultimate syllable *is* reduced'

One can make several types of generalisations. One such generalisation is where the ordering is partial rather than total. Example 4:

'a' = 'wake up'
'b' = 'take a shower'
'c' = 'take a bath'
'd' = 'eat bread'
'e' = 'go to work'

L = {'abde', 'acd', 'aed', 'adb', 'ab'}

In this list, 'b' and 'c' seem mutually exclusive and therefore cannot be mutually ranked (alternatively, it makes no sense to try to rank them, or there are no input data available in which they are ranked). Furthermore, 'd' and 'e' is optional, and there seems a tendency to first do 'a' and later 'b' or 'c', while the ranking of 'd' and 'e' is not clear. We will see below how we can deal with this situation of partial ordering.

Terminology

In the examples mentioned above, we have been dealing with the ordering of actions, properties, and transformations. To uniform the terminology, we will use the term 'constraint' in the sequel of this paper for the objects to be ordered on the basis of input data.

2 Learning the ordering of constraints

The question now is, how the constraints can be organized in a hierarchy or 'grammar' such that this hierarchy (e.g. a linear ranking) optimally reflects the observed orderings such as in the lists L above. The task for the learner (the algorithm) is to deduce a grammar that explains the observed ordering in the input data.

In Optimality Theory (OT), approaches are described to learn this ranking based on incoming data (see e.g. Kager, 1999). In OT, it is assumed that a (fixed) set of constraints is already available to the learner. In some of the approaches, one stipulates that all possible output forms – not only the ones that are observed – are available to the learner, together with violation marks for those constraints that render them sub-optimal. Whether such sub-optimal forms *are* available for the learner, even if they are not presented as such, is a topic of theoretical debate in OT.

In this paper, we will not go into detail on this debate. We will just assume that only the data that are presented are available as learning data, and will focus on and discuss some of the mathematical properties of ranking solutions. Next, we will deal with the question of how these solutions can be found by a sequential learning algorithm, and point to a relation with graph theoretical notions. The result of this exercise is a model that will be referred to as Probabilistic Ranking Optimisation (PRO).

3 Towards a formal approach

We will abstract away from the interpretation of the symbols, and concentrate on the structure of the list L in terms of the statistical properties of the ordering of the constraints. So L can be either a finite list of string data, or be extended with probabilities allotted to each string, or can be an infinite stream of data. It is further assumed that the ordering of the constraints can be total or partial. We mention a number of ways to describe the structure of L.

1. By (probabilistic) grammars. By interpreting L as a language with sentences of words ‘a’, ‘b’, etc., L can be described or approximated by a finite state grammar (FSG), a context-free grammar (CFG), or probabilistic versions of these grammars (e.g. Charniak, 1993). A large number of established algorithms exist to construct the grammar and to update the probabilities of the grammar rules on the basis of L. It will be clear that the set of grammar rules that represents L may become rather cumbersome. For lists with a more complicated ordering structure, the rules become as trivial as the list L itself, for example the rule **rewrite**(S, ‘abde’ | ‘acd’ | ‘aed’ | ‘adb’ | ‘ab’) or a probabilistic version, in the case of example 4. This rule states that the formal sentence symbol S can be rewritten into one of the five strings. The grammar approach is useful to factor out groups of constraints that behave as ‘mutually disjoint’. We will address the probabilistic grammar approach below.
2. By (probabilistic) ordering. Probabilistic ordering (ranking) is used in recent developments in OT (for example in the Gradual Learning Algorithm (GLA), see Boersma & Hayes, 2001). The basic idea is, for a totally ordered ranking system, to have each action ‘x’ associated with a location pos(‘x’) on a straight line, such that pos(‘a’) < pos(‘b’) means that ‘a’ is likely to occur before ‘b’. The larger the distance between pos(‘a’) and pos(‘b’), the more likely ‘a’ occurs before ‘b’, and reverse: if pos(‘a’) > pos(‘b’), ‘a’ is likely to happen after ‘b’. In the sequel, we will follow this approach and further elaborate on it. In this paper, probabilistic ranking will be notated by the rule **rewrite**(S, probabilistic ordering{‘a’, …, ‘z’}), meaning a rewrite of the formal sentence symbol S into a probabilistic string using the symbols ‘a’ to ‘z’, i.e. an probabilistic ordering on the constraints ‘a’ to ‘z’. The statistics of the ordering evidently depends on the model parameters such as pos(‘a’), but these parameters will be omitted here not to burden the notation.

One of the numerical *models* that implements probabilistic ranking makes use of probability distributions, such that each action ‘x’ corresponds with some probability

distribution D (in general, D will depend on x). D is a probability distribution on the set of real numbers. The probability that 'a' occurs before 'b' – henceforth denoted P('a' < 'b') – is then modelled as

$$P(\text{'a'} < \text{'b'}) = P(X_{\text{'a'}} < X_{\text{'b'}} \mid X_{\text{'a'}} \text{ and } X_{\text{'b'}} \text{ stochastic variables associated with } D_{\text{'a'}} \text{ and } D_{\text{'b'}}, \text{ resp.}) \quad (1)$$

This definition makes sense for all sorts of probability distributions D. The two distributions $D_{\text{'a'}}$ and $D_{\text{'b'}}$ need not be of the same 'shape' or from the same family of distributions. When the two distributions do not overlap, e.g. when $D_{\text{'a'}}$ is entirely located at the left side of $D_{\text{'b'}}$, then P('a' < 'b') = 1, which means that 'a' always occurs before 'b' ('a' always outranks 'b'). The involved probability evaluation takes a particularly simple form in the case where the distributions have a simple mathematical form, such as a triangle shape, a laplacian or a gaussian. The choice for a particular form for the distributions can be based on additional knowledge about an underlying evaluation model, or on arguments related to mathematical elegance. For example, in the case of gaussians, the probability P('a' < 'b') will never completely vanish, nor will it ever be exactly equal to 1, properties which may be attractive from a probabilistic point of view and are often essential for learning.

In the sequel, we will use gaussian distributions to show a number of properties of probabilistic ranking. In the gaussian case, the probability P('a' < 'b') can be expressed just in terms of two means and two standard deviations. It can easily be shown that, if the gaussian distribution $G_{\text{'x'}}$ is specified by its mean $\mu_{\text{'x'}}$ and standard deviation $\sigma_{\text{'x'}}$, eq. (1) is simplified to eq. (2):

$$P(\text{'a'} < \text{'b'}) = IN(-\mu/\sigma), \text{ with}$$
$$\mu = \mu_{\text{'a'}} - \mu_{\text{'b'}} \quad (2)$$
$$\sigma^2 = \sigma_{\text{'a'}}^2 + \sigma_{\text{'b'}}^2, \ \sigma \text{ positive or zero}$$

where IN(*x*) denotes the integral from –infinity to *x* of the normal distribution. Observe that IN(γ) + IN(-γ) = 1 for all values of γ. After reorganisation of eq. (2), eq. (2a) follows:

$$(\mu_{\text{'a'}} - \mu_{\text{'b'}}) = \gamma_{\text{'ab'}} \, \text{sqrt}(\sigma_{\text{'a'}}^2 + \sigma_{\text{'b'}}^2) \quad (2a)$$

in which $\gamma_{\text{'ab'}}$ denotes the unique solution such that P('a' < 'b') = IN($-\gamma_{\text{'ab'}}$). In the case where the variance is fixed to 1, (2a) simplifies to (2b):

$$(\mu_{\text{'a'}} - \mu_{\text{'b'}}) = \gamma_{\text{'ab'}} \, \text{sqrt}(2) \quad (2b)$$

Observe that eqs. (2) and (2a) also hold in the more general case, in which the distributions $D_{\text{'x'}}$ are not necessarily gaussian. In such a general case, the relation between $\gamma_{\text{'ab'}}$ and P('a' < 'b') is different from the gaussian case. In the case that $D_{\text{'x'}}$ vanishes nowhere (which is true for gaussians and many more distributions), the constant $\gamma_{\text{'ab'}}$ is *one-to-one* related to the probability P('a' < 'b') via a more general function F:

$$P(\text{'a'} < \text{'b'}) = F(-\gamma_{\text{'ab'}})$$

in which F only depends on the general type of distributions.

Obviously, by varying the model parameters in the distributions, one can change the mismatch between the observed orderings in the incoming data in L on the one hand, and the predicted orderings (by using eq. 1) on the other. For example, if 'a' is always found to outrank 'b', a model with $\mu_{'a'} < \mu_{'b'}$ will yield a better match than a model in which $\mu_{'a'} > \mu_{'b'}$. For a given L, specific choices of the distributions will yield an optimal match. The existence of such 'optimal' solutions when gaussians are used, and the involved learnability aspects, will be the focus of the next section.

4 Existence of rankings

Two constraints

In this section, we focus on the question whether ranking solutions exist in terms of means and standard deviations for an arbitrary incoming data L. We will distinguish two probabilistic ranking optimisation models: PRO-1 in which every $D_{'x'}$ is characterized by one parameter (the mean), and PRO-2, in which every $D_{'x'}$ has two parameters (mean and variance).

We will start by examining the simplest non-trivial case in which we have only 2 constraints to be ranked. In this case, there is only one probability, viz. P('a' < 'b'), to be modelled. In PRO-2, the potential solution space is four-dimensional, since there are two constraints, each having two model parameters. The modelling of P('a' < 'b') yields a non-trivial (non-linear) relation between the four parameters $\{\mu_{'a'}, \sigma_{'a'}^2, \mu_{'b'}, \sigma_{'b'}^2\}$, which is specified by eq. (2a). This leaves three degree of freedom in the solution space. Among these four parameters there are two additional but *trivial* relations, one related to a *shift* of all the means and the second related to a common *scaling* factor for the means and standard deviations. Taking away also these two trivial degrees of freedom, one 'genuine' degree of freedom is left, which indicates that, if we use means and variances for each gaussian, we end up with a one-dimensional model space. This means that there is not a single solution but a one-dimensional family of solutions: the solution in PRO-2 is underspecified by just specifying P('a' < 'b'). The result means that, *if* any solution exists, the genuine solution space around that solution *in general* is one-dimensional.

The value of the dimension (viz. 1) itself is independent of the actual value of P('a' < 'b'). This in turn means that a tracking of the 'recent' statistics in the input data is possible but underspecified as well, since each neighbouring solution is also embedded in its own solution space of dimension 1.

This 'freedom' in the solution space can be used to put more restrictions on the solution – if one additional restriction is imposed, the solution becomes unique. In the Gradual Learning Algorithm (GLA), Boersma & Hayes (2001) fix one of the means and set both variances equal to one, thereby reducing the solution space to dimension 0 (i.e. exactly one solution, if it exists). This is actually the solution according to PRO-1. In summary, it follows that in the two-constraints case, a ranking solution (with or without fixed standard deviations) always exists, the solution in GLA or PRO-1 being unique.

Three constraints

PRO-2. When more than two constraints are involved, things get slightly more difficult. In the case of three constraints ‘a’, ‘b’, ‘c’ that are to be totally ordered, the six parameters $\{\mu_{‘a’}, \sigma_{‘a’}^2, \mu_{‘b’}, \sigma_{‘b’}^2, \mu_{‘c’}, \sigma_{‘c’}^2\}$ are determined by up to three (3*(3-1)/2) relations associated to the three observed probabilities P(‘a’ < ‘b’), P(‘b’ < ‘c’) and P(‘a’ < ‘c’). If $\gamma_{‘ab’}$ is chosen such that P(‘a’ < ‘b’) = IN($-\gamma_{‘ab’}$), and similarly for $\gamma_{‘bc’}$ and $\gamma_{‘ac’}$ (these parameters γ are now uniquely determined), we obtain a system of three equalities, each of them fully comparable with eq. (2a) but now corresponding to the relation between ‘a’ and ‘b’, ‘b’ and ‘c’, and ‘a’ and ‘c’, respectively:

$$\begin{aligned}(\mu_{‘a’} - \mu_{‘b’}) &= \gamma_{‘ab’}\ \mathrm{sqrt}(\sigma_{‘a’}^2 + \sigma_{‘b’}^2)\\ (\mu_{‘b’} - \mu_{‘c’}) &= \gamma_{‘bc’}\ \mathrm{sqrt}(\sigma_{‘b’}^2 + \sigma_{‘c’}^2)\\ (\mu_{‘c’} - \mu_{‘a’}) &= \gamma_{‘ca’}\ \mathrm{sqrt}(\sigma_{‘c’}^2 + \sigma_{‘a’}^2)\end{aligned} \qquad (3)$$

These three relations leave three (6 – 3) degrees of freedom. As before, the remaining three dimensions include two trivial degrees of freedom: a shift of the means and a scaling factor for means and standard deviations, so there is only one genuine degree of freedom left.

In this case, however, there is not always an exact solution in terms of $\{\mu_{‘a’}, \sigma_{‘a’}^2, \mu_{‘b’}, \sigma_{‘b’}^2, \mu_{‘c’}, \sigma_{‘c’}^2\}$. That means that there is a combination for P(‘a’ < ‘b’), P(‘b’ < ‘c’) and P(‘a’ < ‘c’) that cannot be modelled by this probabilistic ranking approach. It can be shown that an exact solution exists only in the following two cases (a) and (b):
(a) when the inner product

$$((\gamma_{‘ab’}, \gamma_{‘bc’}, \gamma_{‘ca’})’, P) \qquad (4)$$

takes different signs on the set $\{P\} = \{(1\ 1\ 0)’, (1\ 0\ 1)’, (0\ 1\ 1)’\}$. In words: within the set of three numbers $\{\gamma_{‘ab’} + \gamma_{‘bc’}, \gamma_{‘ab’} + \gamma_{‘ca’}, \gamma_{‘bc’} + \gamma_{‘ca’}\}$, at least one must be strictly positive and at least one must be strictly negative, in order to have an exact solution for the ranking with the specified P(‘a’ < ‘b’), P(‘b’ < ‘c’), and P(‘a’ < ‘c’).
(b) when all γ’s are zero, which means no preference for any symbol, i.e. P(‘a’ < ‘b’) = P(‘b’ < ‘c’) = P(‘a’ < ‘c’) = ½. Modulo shift and scaling, the solution space is essentially two-dimensional.

PRO-1. In the case when all variances are fixed to 1 in the 3-constraints case, it follows from eq. (3) that there is no solution unless coincidentally $\gamma_{‘ab’} + \gamma_{‘bc’} = \gamma_{‘ac’}$ (or equivalently $\gamma_{‘ab’} + \gamma_{‘bc’} + \gamma_{‘ca’} = 0$). In that case, the genuine solution space has dimension 0 (apart from a trivial shift, all means are determined). In general however, such an equation between the γ’s does not hold, since P(‘a’ < ‘c’) is not specified by P(‘a’ < ‘b’) and P(‘b’ < ‘c’). This can be easily inferred from table I. The table shows that P(‘a’ < ‘c’) = #(‘a’ < ‘c’)/|L| can take any value between 0 and P(‘a’ < ‘b’) + P(‘b’ < ‘c’), by appropriately defining the probabilities of the six strings ‘abc’, …, ‘cba’.

It follows that the probabilistic ranking using single gaussians per symbol is not able to model all possible rankings between three symbols when the variances are forced to 1. In the gradual learning algorithm (GLA) described by Boersma & Hayes (2001), all variances are chosen equal to unity, despite the involved loss in mathematical modelling power, and they give an argument to do so.

Table I. By appropriately manipulation of the probabilities of the strings ‘abc’ to ‘cba’ in the input, P(‘a’ < ‘c’) can take arbitrary values between 0 and P(‘a’ < ‘b’) + P(‘b’ < ‘c’).

	#(‘a’<’b’)	#(‘b’<’c’)	#(‘a’<’c’)
abc	*	*	*
acb	*	-	*
bac	-	*	*
bca	-	*	-
cab	*	-	-
cba	-	-	-

Boersma & Hayes relate the variance of a constraint to a property of the (human) evaluation system, rather than to the constraint itself. Their argument is that variances reflect the noise inherent to the *evaluation* of the features that are input for the probability distributions, and since the evaluation phase precedes the decision, it must be the same for each constraint.

In this paper, we do not adopt this view. Although part of the variance is related to evaluation noise, we do not conclude that all variances must *therefore* be equal. More precisely, evaluation noise refers to the fact that repeated presentation of the same stimulus yields different perceptual effects (among the extensive literature see Ashby, 1992; Ashby & Alfonso-Reese, 1995), rather than to direct claims about the shape of statistical distributions. Furthermore, with equal covariance matrices per class, the boundaries between all classes would be linear, which is in general an unnecessary strong assumption in classification theory.

In addition, noise plays an essential role in the theory on decision rules and perceptual representations (Maddox & Bogdanov, 2000), and it is a priori possible that a certain constraint is more susceptible to noise than another constraint. We therefore take the position that the variance can be regarded as a genuine, intrinsic property of a constraint, which *may* include noise effects from the evaluation channel, but which eventually fully specifies its vulnerability among neighbouring constraints. We however agree that, in the PRO-2 model, the variances are ‘overloaded’ in the sense that they are used to both cover the intrinsic variation in the input data as well as to model a noisy evaluation system.

Unequal variances are essential if one wants to model

L = {‘bac’, ‘abc’, ‘acb’}

in which ‘a’ always outranks ‘c’. (Boersma & Hayes provide a similar example). With different variances, an exact modelling of this list L is possible.

But even with *unequal* variances, the probabilistic model is not capable of explaining the statistics of any list L with three constraints. For example, if L takes a cyclic form {‘abc’, ‘bca’, ‘cab’} (with probability 1/3 for each element), the solution is degenerated to $\mu_{‘a’} = \mu_{‘b’} = \mu_{‘c’}$ = a constant. L is only a subset of patterns L’ that are explained by the probabilistic ranking:

L’ = {‘abc’, ‘bca’, ‘cab’, ‘acb’, ‘bac’, ‘cba’}

with equal probability (1/6) for each of the strings. So the probability model accepts many more patterns than it has observed. Mathematically, this ranking problem is solvable when the probabilistic model would be extended to allow the ranking of strings with unequal lengths, such as P('ab' < c'), P('a' < 'cb'), etc. The gradual learning algorithm (GLA) and the PRO models, however, do not take into account these higher-order probabilities. (In section 9, the background of this 'cyclic' situation is explained in a more geometrical way.)

We again emphasize that the present treatment deals with the existence of an *exact* solution. In practice, 'nearly optimal' solutions will exist in many cases. These nearly optimal solutions are easiest to derive in the PRO-1 model. For example, in the case of three constraints only, we have to solve the following set of equations (5):

$$\begin{aligned} (\mu_{'a'} - \mu_{'b'}) &= \gamma_{'ab'}\ \mathrm{sqrt}(2) \\ (\mu_{'b'} - \mu_{'c'}) &= \gamma_{'bc'}\ \mathrm{sqrt}(2) \\ (\mu_{'c'} - \mu_{'a'}) &= \gamma_{'ca'}\ \mathrm{sqrt}(2) \end{aligned} \qquad (5)$$

where the γ's are fully determined by the observed probabilities. If their sum equals 0, the PRO-1 model can exactly match the observed ranking probabilities. However, if the sum is not exactly equal to 0, the model can only approximate the observed patterns, and in such a case it is essential to use a precise definition of 'approximate'. The literature on optimisation provides numerous methods to search a model that maximizes a 'match' with the data (e.g. by optimising the likelihood of the data, by least squares minimization, or by expectation maximisation), but the choice of the optimisation is also a matter of elegance and feasibility with respect to the targeted application domain (more about this in section 6).

Four and more constraints

PRO-2. In case of four constraints 'a', 'b', 'c' and 'd' that are totally ordered, the 6 probabilities P('x' < 'y') lead to 6 equations of the same type as eq. (2a). The number of variables is 8, which means that the remaining dimension is 2 (which include the shift and the scaling). So, in essence, there is just *one* solution, if it exists. In the same way as in the three-constraint case, it is possible to give a number of *necessary* conditions for the existence of the solution, if one considers the case of *total* ordering. These conditions are based on the 4 different cycles of length 3 in the complete graph K_4 with 4 vertices and 6 edges. If the vertices of this graph are labelled by 'a', 'b', 'c' and 'd', the four 3-cycles read {'a', 'b', 'c'}, {'a', 'b', 'd'}, {'a', 'c', 'd'}, and {'b', 'c', 'd'} (each to be considered cyclically). Within each of these subsystems, a set of equations hold similar to eq. (3). When a solution of the entire system K_4 exists, it must at least hold on the 4 3-cycles, i.e. the four associated equations (4) must each admit a solution.

PRO-1. The six probabilities P('x' < 'y') lead to 6 equations of the same type as eq. (2b). There are 4 free parameters, which have to satisfy 6 equations, which in general means that there is no solution possible. Solutions exist in special cases, where $\gamma_{'xy'} + \gamma_{'yz'} + \gamma_{'zx'} = 0$, where {x, y, z} run over all triangle vertices in the complete graph K_4. In that case, there is only one genuine solution.

Table 2. Overview of the results for PRO-1 (top) and PRO-2 (bottom). The assumed ranking is totally ordered.

n	#var	#condit.	#dim	genuine solution exists?
2	2	1	0	unique
N>2	N	N(N-1)/2	<0	special cases only

n	#var	#condit.	#dim	genuine solution exists?
2	4	1	1	1-dim
3	6	3	1	1-dim
4	8	6	0	unique
N>4	2N	N(N-1)/2	<0	special cases only

The results for totally ranked systems are summarized in Table 2.

When N is large enough, the dimension of the solution space becomes negative, which implies that a general exact solution does not exist. However, a particular exact ranking solution *may* exist for special choices for the probabilities P(‘x’ < ‘y’). This exact ranking solution turns into an approximate solution when the probabilities deviate from the special setting. In section 9 we will discuss this ‘approximate’ case in more detail.

Partial solutions

PRO-2. In general, when N constraints have to be totally ranked, then N(N-1)/2 equalities of the form of equation (2) have to be fulfilled. Each of these equalities subtracts one degree of freedom from the solution space (of which the dimension is 2N, twice the number of constraints, in the case of no ranking equations at all). If the ranking is partial rather than total, then we do not have the full number of N(N-1)/2 equalities but just M of them (where M < N(N-1)/2).Taking into account the two trivial degrees of freedom, we have for the eventual genuine dimension (dim) of the solution space:

$$\dim = 2N - M - 2 \qquad (6)$$

A non-negative value of dim does not *guarantee* the existence of a solution. It only means that, *if* a solution exists, *then* the solution space around the solution has dimension dim. Necessary conditions for the existence of solutions are cumbersome to produce; see below.

As an example, we consider the case in which four constraints ‘a’ to ‘d’ are to be ranked where only the probabilities along one 4-cycle {‘a’, ‘b’, ‘c’, ‘d’} in K_4 are specified, i.e. we only know P(‘a’ < ‘b’), P(‘b’ < ‘c’), P(‘c’ < ‘d’), and P(‘d’ < ‘a’). In this case N = 4 and M = 4, so

$$\dim = 2*4 - 4 - 2 = 2$$

which means that the genuine solution space has dimension 2. If additionally P(‘b’ < ‘d’) and P(‘a’ < ‘c’) are specified, the dimension reduces to 0, as we have already seen before.

PRO-1. In case of N constraints, the genuine solution space without any imposed equations of the form (2b) has dimension N – 1. Each equation of the form (2b) decreases the dimension by 1. However, by doing so, we discount too much for cycles, and in order to correct that we must increase the dimension by 1 for every face

except for the default ‘unbounded face’ when embedding G in the plane (see section 7 for details). With M equations to be satisfied, the resulting genuine dimension reads

$$\begin{aligned} \dim &= N - 1 - M + (F - 1) \\ &= F - M + N - 2 \end{aligned} \qquad (7)$$

Optional constraints

A special case occurs if we have *optional* constraints in the data. We will briefly discuss this situation without going into detail here. In example (4), we deal with 5 symbols of which ‘b’ and ‘c’ are incomparable, and in which ‘c’, ‘d’ and ‘e’ are optional (see figure 1).

L = {‘abde’, ‘acd’, ‘aed’, ‘adb’, ‘ab’}

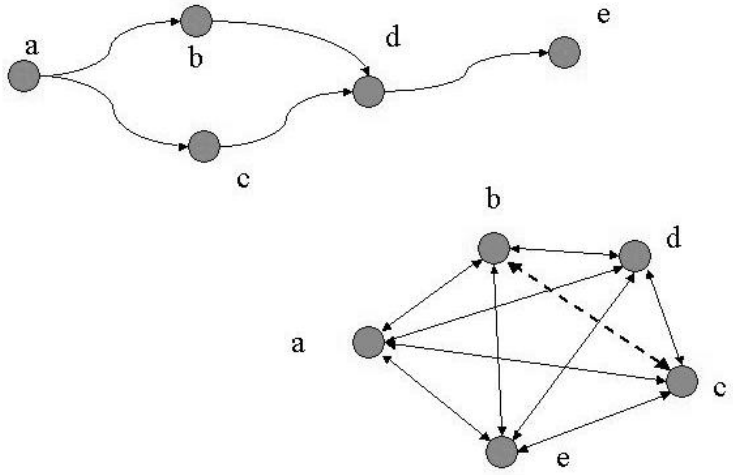

Figure 1. Ranking of five constraints, ‘b’ and ‘c’ mutually independent.

The top in figure 1 shows the lattice associated to example 4. The five constraints do not move along one line, but ‘b’ and ‘c’ move independently – i.e. without mutual ranking limitations – along parallel paths. The structure of the lattice follows from the fact that all symbols can be simultaneously in an element of L, except for ‘b’ and ‘c’. That means, that of the 10 possible relations (similar to eq. 2), only 9 are effective (figure 1 bottom right). But in this example, we have to take optionality into account, since ‘b’ and ‘c’ are not mandatory. This can be modelled as follows (compare eq. 1):

$$\begin{aligned} P(\text{‘a’} < \text{‘b’}) = {} & P(X_{\text{‘a’}} < X_{\text{‘b’}} \mid X_{\text{‘a’}} \text{ and } X_{\text{‘b’}} \text{ stochastic variables} \\ & \text{associated with } D_{\text{‘a’}} \text{ and } D_{\text{‘b’}} \mid \text{both ‘a’ and ‘b’ are produced}) \\ & * P(\text{‘a’ \& ‘b’}) \end{aligned} \qquad (8)$$

where P(‘a’ & ‘b’) denote the a priori probability of the applicability of *both* the constraints ‘a’ *and* ‘b’ in an arbitrary input datum. So, the probability of ‘a’ outranking ‘b’ now has become conditionally dependent on the applicability of ‘a’ and

'b'. The probability P('a' & 'b') can be trained from the input data. (The problem of estimating these probabilities resembles the estimation of hidden parameters in hidden markov modelling, which is part of standard missing data theory, in which the statistics of not directly observable states is dealt with.) Further simplifying, one might reduce the probability P('x' & 'y') into two factors P('x') and P('y'), representing the probabilities of observing the individual constraints 'x' and 'y' in the input. Evidently, such a factorisation is only correct in the case of independence of the occurrences of 'x' and 'y'.

In OT, one has introduced the notion of 'strata', i.e. collections of constraints without internal ranking, to deal with incomparable constraints. The difference between the notion of 'stratum' and the present treatment is that, although 'b' and 'c' are mutually incomparable (in example 4), both constraints can be compared e.g. to 'a', with possibly different outcomes. So, although 'b' and 'c' cannot be mutually compared, their ranking behaviour towards a third constraint is different.

5 Towards a combination of a probabilistic grammar and probabilistic ranking

So far, we considered ranking of symbols in a lattice, of which the structure is determined by the set of equations (3). Each of these equations is based on a constraint-to-constraint comparison. In certain cases, this is evidently not a satisfying way to describe the structure of L. For example, consider the two lists

L_1 = {'abcd', 'bacd', 'abdc', 'abdc', 'abdc'}
L_2 = {'cdeab', 'abcde', 'cdeab', 'abcde', 'abcde'}

The structure of L_1 can be described (to a very good approximation) by a total ranking of the four constraints, i.e. by a grammar

rewrite(S, probabilistic ordering {'a', 'b', 'c', 'd'})

but it makes sense to consider an alternative description such as (A and B denoting non-terminals):

rewrite(S, 'A B')
rewrite(A, probabilistic ordering {'a', 'b'})
rewrite(B, probabilistic ordering {'c', 'd'})

L_2 cannot be properly described by **rewrite**(S, probabilistic ordering {'a', 'b', 'c', 'd', 'e'}) since any of such rankings would incorrectly explain 'cabde', 'cdabe' or 'acdeb' with positive chance. The use of a CFG yields a correct description, as follows:

rewrite(S, probabilistic ordering {'A', 'B'})
rewrite(A, 'ab')
rewrite(B, 'cde')

From these examples, it follows that a combined use of a probabilistic finite state grammar (PFSG) or a probabilistic context free grammar (PCFG) and probabilistic ranking can provide a powerful tool to describe the statistical properties in L. All the

involved probabilities can be learned from data: the probabilities in a PCFG can be trained on the basis of the data in L by the *inside-outside algorithm* (Charniak, 1993), while the parameters of the probabilistic ordering can be learned by GLA or a similar soft-ranking algorithm.

This type of combined description leads to the question what the differences are between the following two descriptions D_1 and D_2:

D_1:
rewrite(S, probabilistic ordering {'A', 'B', 'C', …'Z'})
rewrite(A, probabilistic ordering {'a_1', …, 'a_{Na}'})
rewrite(B, probabilistic ordering {'b_1', …, 'b_{Nb}'})
…
rewrite(Z, probabilistic ordering {'z_1', …, 'z_{Nz}'})

and

D_2:
rewrite(S, probabilistic ordering {'a_1', 'a_2', 'a_{Na}', 'b_1', …, 'z_{Nz}'})

The answer is, that if the constraint groups {'a_i'} and {'b_j'} do not interlace (so, all 'a's rank out all 'b's, or vice versa), and idem for all other combinations of constraint groups, then D_2 can be put in the form of D_1. The outranking is never strict since there might always be a small probability – depending on the form of the probability distributions used – of observing an extremely rare ordering, so theoretically the disjunction of constraint groups is never entirely satisfied. However, in these cases, D_1 and D_2 can be *practically* equivalent. Conversely, if in D_1 the rule

rewrite(S, probabilistic ordering {'A', 'B', 'C', …'Z'})

practically outputs a fixed ordering of 'A', …, 'Z', then D_1 can be put in the form of D_2. In general, however, D_1 and D_2 model different types of statistics in L. The training of D_2 can be performed via the soft-ranking algorithm, while the training of D_1 involves a *nesting* of soft rankings.

6 Learnability

Suppose a set of input 'surface' forms and a set of constraints are given. What strategies can the learner use to discover the proper ranking of the constraints? The study about learnability of ranking is initiated by Tesar & Smolensky (1993). Their hard-ranking learning algorithm (recursive demotion) demotes rankings in such a way that the highly ranked constraints become less violated. For each underlying representation, a 'module' GEN generates an infinite number of output surface realisations (candidates). These candidates are evaluated against a set of constraints (by the module EVAL). In the standard OT, constraints are assumed to be universal (language-independent) and specified by some universal grammar. Constraints are ranked in a language-specific hierarchy (a grammar). The winning ranking is the one with the least serious violations with respect to the data. The ultimate goal is to

discover a general (universal) set of constraints shared by all languages. The question whether a universal set of constraints exists is not settled.

The hard-ranking algorithm demonstrates that it is possible to deduce rankings of constraints on the basis of incoming surface data. It is based on a number of assumptions. First, it is assumed that all data presented to the learner are grammatically correct. Also it is alleged that the learning algorithm can access the correct underlying representation, besides the raw surface form: the hypothesised underlying forms are to be deduced from the surface form and the constraint hierarchy. The observed violations produce the crucial information for updating the constraint ranking, since constraints that are violated *must* be strictly outranked by some other constraint. About the *totality* of the ranking, Prince & Smolensky (1993, p. 51) observe that this is not an a priori assumption of OT.

The soft-ranking model implemented in e.g. the Gradual Learning Algorithm (GLA) shows that the *probabilistic* ranking of constraints is possible using the same type of structured data. The initial state in most of these algorithms is one in which all constraints are unranked with respect to one another. Boersma (1997) has shown that the GLA is able to reflect the statistics found in the training data, for a large number of different phonological patterns.

Learnability from a mathematical viewpoint

We now consider learnability from a more mathematical perspective. In a probabilistic ranking model, the locations of the probability distributions specify the probability of observing the strings 'abcd', 'dcab' etc. For example, if all four distributions have equal means, all 24 4-character strings are equally probable (each having a probability1/24), regardless the standard deviations. And if $\mu_{\text{'a'}} < \mu_{\text{'b'}} < \mu_{\text{'c'}} < \mu_{\text{'d'}}$, 'abcd' is preferred over all other orderings. So, by manipulating the distributions (by altering the model parameters, i.e. the means and standard deviations), one can minimize the *statistical* mismatch between the predicted distribution of strings (on the basis of the probabilistic model) on one hand and the observed string distribution in the list L on the other. This can be formulated as the optimisation of the *likelihood* $P(L \mid M)$ of observing L given the model M, where $M = M(\lambda)$ depends on a parameter set (denoted by λ). The maximum likelihood solutions can be found in a number of numerical ways, of which so-called Expectation Maximization (EM) algorithms are a well-known family (e.g. Dempster et al., 1977; Lee & Gauvain, 1996). EM algorithms, which are commonly used in the training of Hidden Markov Models in automatic speech recognition, consist of two steps (an E-step and a subsequent M-step) which are iterated a number of times until convergence takes place. In general, EM algorithms have a *sequential* variant in which the learning data are presented in the form of a stream and the model parameters are updated after each input datum so as to optimise a *running* maximum likelihood criterion. There is a massive amount of literature on adaptive learning algorithms that update the model parameters for each incoming input datum such that some target function is minimized on the long term. The target function expresses the mismatch between the statistics generated by the model and the observed 'most recent' statistics in the input stream (well documented examples of such learning algorithms are: sequential learning, reinforcement learning, Q-learning, the Palo-algorithm; see Kearns & Vazirani, 1994; Greiner, 1996; Cowel et al., 1996). As with all optimisation algorithms, there is no general guarantee to avoid being stuck in a local optimum.

In principle, sequential learning algorithms start with a random distribution and, after a number of input data from the input stream, they 'tune in' on the actual distribution in the stream. A number of learning algorithms assume that it is possible – at least in principle – to access and update *all* model parameters on the basis of one input datum. Other learning algorithms modify the model more locally: only certain parameters are updated per input datum. In the one-dimensional GLA, this means that the 'local' type of algorithms demote (= shift to the right) one or a few non-matching distributions ('violation'), or promote the location of one or a few matching distributions ('satisfaction'). Seen from the mathematical point of view, it is evident that the ranking by e.g. soft-ranking algorithms can describe the recent statistics in the input, since that is a general property of an extensive family of learning algorithms to which soft-ranking algorithms – as far as they are interpreted as maximum likelihood optimisation techniques – belong.

The question which type of learning (soft-ranking, or other variants) applies in a certain case is a matter of the plausibility of a model embedding into a certain paradigm, rather than a matter of statistical matching qualities alone. For example, for issues in phonology, where issues concerning *acquisition* of phonological patterns play a role, the structural complexity of the learning algorithm is a matter of concern. These embedding issues – which are related to interpretation of the model – are not addressed in this paper.

Speed of convergence to optimality.

Optimality is usually an asymptotic result, and so convergence *speed* itself is a poorly defined measure. More interesting are the speed of convergence to near-optimality and the shape of the learning curve. This learning behaviour is studied in a more general setting which is called the Probably Approximately Correct (PAC) framework (e.g. Haussler, 1990). The PAC framework takes as a starting point that a leaner is equipped with a class of possible classifiers (e.g. decision trees, or multiplayer perceptrons). A number of labelled training instances is presented to the learner; the learner uses these training data to identify a certain specific classifier from the given class. Each classifier in the class will show an error (the probability to incorrectly classify the input). For certain classes of classifiers it is possible to guarantee that the classifier found by the learner will have – with high probability – a small error. PAC also provides information about the minimal length of the learning period in order to have a probabilistic guarantee on a certain performance on a (independent) test set. Optimisation algorithms show in general a trade-off between the speed of training and the performance on a test set. Another useful performance measure (known as *regret*) is the expected decrease in performance on a test set, due to the execution of the learning algorithm instead of behaving optimally directly from the start.

7 Relations with graph theory

There exists an interesting relation between constraint ranking and graph theory. To explain this relation, we need two concepts related to a graph, *face* and *genus*.

If a graph G is drawn on ('embedded in') a surface, the V vertices and E edges decompose the surface into F polygonal regions. The surface can be the plane, the sphere, a torus (the surface of a donut), etc. (the embedding in a plane or in a sphere is not essentially different). The surface can always be chosen such that there are no

illegal crossings of edges ('self-crossings') and furthermore that every region is a *face*. A face is a region that is *simply connected*, i.e. it can be continuously shrunk to a single point without leaving the surface. As an example we consider the complete graph K_3, which consists of three vertices connected by three edges. Three edges is the maximum number to connect three vertices, hence the adjective 'complete'. This graph K_3 embedded in the plane or the sphere has two faces: the 'inside' and the 'outside'. The graph K_4 embedded in the sphere has 4 faces (see figure 2). The graph $K_4 + K_4$ (the graph that consists of two graphs K_4 that are connected with one single edge) on the sphere has 7 faces. The complete graph K_5 cannot be embedded in the plane without self-crossings (in jargon: the graph is not *planar*), but it can be properly embedded on the torus in which case it has 5 faces (see figure 3).

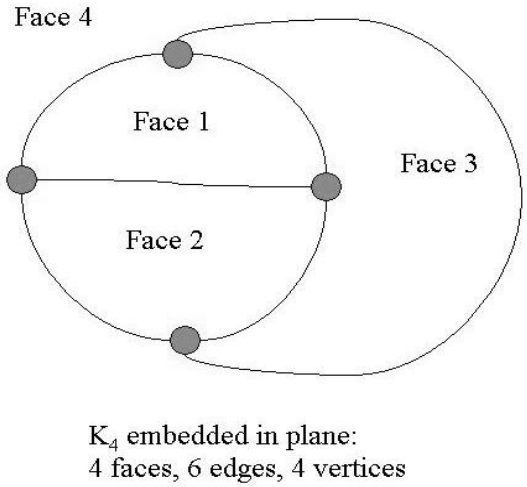

Figure 2. The complete graph K4 embedded in the plane (or on a sphere).

The *genus* of a graph, usually denoted by Γ, is the *minimum* number of 'handles' attached to the sphere that are necessary to embed the graph without any self-crossings. Graphs such as K_3 and K_4 can be drawn on the sphere (in the sphere) without self-crossings: both $\Gamma(K_3)$ and $\Gamma(K_4)$ are therefore equal to 0. Since the torus is essentially a sphere with just one 'handle' and the graph K_5 can be drawn on a torus without self-crossings, $\Gamma(K_5) = 1$.

The number of components of a graph is (loosely speaking) the number of parts of the graph that are not connected with one another. For example, the number of components of the graph consisting of just *k* vertices is *k*. When the graph is connected, the number of components equals 1.

Let the graph G now be defined by the constraints as the vertices and the imposed comparability relations as the edges. Then the dimension of the genuine solution space of the ranking problem in PRO-2 depends on the structure of G as follows. We define the following parameters (# denotes 'number of'):

#N: the number of vertices in G
#EC: the number of components of G, minus one

(i.e. the number of 'extra components')
#F: the number of faces of G
#E: the number of edges of G

Then the genuine dimension of the ranking solution reads:

$$\dim = \#N + \#EC - \#F - 2\Gamma(G) \qquad (8)$$

In words, eq. (8) states the following. Every vertex (i.e. each constraint) increases the dimension by one. Every additional graph component also increases the dimension by one. Contrary, every face reduces the dimension by one; and handles are penalized by a reduction of 2. In short:

Constraints and components facilitate solutions;
Faces and handles hamper solutions.

The equation (8) can be derived from Euler's polyhedral equation. For a planar graph this equation takes the simple form:

$$\#F - \#E + \#N = 2$$

For a general, non-planar not-connected graph without self-crossings, one obtains (e.g. Coxeter, 1969)

$$\#F - \#E + \#N = 2 - 2\Gamma(G) + \#EC \qquad (9)$$

From eq. (9), it follows that,

$$\#N - \#E - 2 = -2\Gamma(G) + \#EC - \#F$$

Combining this result with eq. (6), one gets

$$\begin{aligned}\dim &= 2\#N - \#E - 2 \\ &= \#N + (\#N - \#E - 2) \\ &= \#N - 2\Gamma(G) + \#EC - \#F \\ &= \#N + \#EC - \#F - 2\Gamma(G)\end{aligned}$$

We give a few examples. $G = K_4$ implies #N = 4, #EC = 0 (there is just one component), #F = 4, #H = 0, which yields dim = 0, as observed earlier.

In the same way, we can deal with K_5. We already know that in general K_5 cannot be ranked. This also follows from eq. (8). Because $\Gamma(K_5) = 1$, we get:

$$\dim = 5 + 0 - 5 - 2 = -2 < 0$$

Two *isolated* copies of K_4 yield dim = 8 + 1 – (3+3+1) – 0 = 2 > 0. These remaining dimensions are precisely the shift and scaling for the second group of constraints, after choosing them for the first K_4.

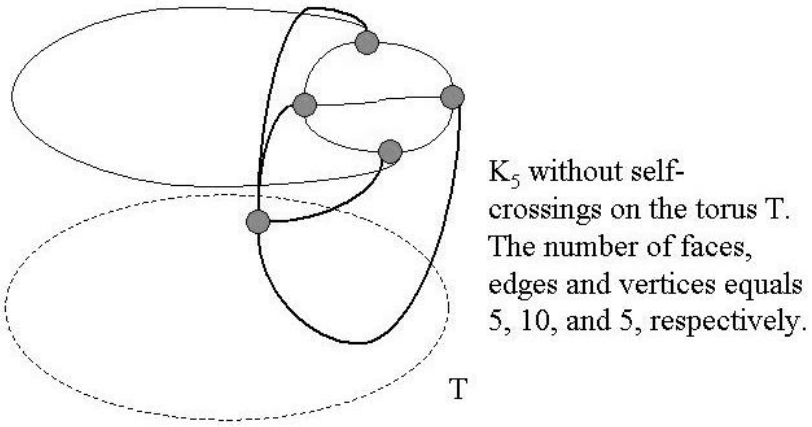

Figure 3. The complete graph K_5 on the torus T without self-crossings. The dashed ellipse represents one of the axes of the torus.

$K_4 + K_4$ (the graph that consists of two copies of K_4 connected with one single edge) yields a dimension of

$$\dim = 8 + 0 - (3+3+1) - 0 = 1 > 0$$

PRO-1. Let G denote the graph corresponding to the N constraints and M equalities. The genuine solution space without any imposed equations of the form (2b) has dimension N – 1. As observed earlier, each equation of the form (2b) decreases the dimension by 1. Due to the linear dependencies in the left hand side of eq. (5), which correspond to the faces of G, we discount too much by doing so. Therefore dim is increased by 1 for every face except for the default 'unbounded face' when embedding G in the plane. Therefore the resulting genuine dimension reads

$$\dim = N - 1 + M + (F - 1)$$

We can simplify this equation by using Euler's polyhedral equation (9), which leads to

$$\dim = F - M + N - 2 = -2\Gamma(G) + \#EC$$

In words*: components facilitate solutions; handles hamper solutions*. If the number of extra components #EC equals 0, then the solution, if it exists, is unique. For N constraints without any restrictions on the probabilities, $\Gamma(G) = 0$ and therefore dim = #EC = N – 1. (This is also obvious without reference to graph theoretic notions.) If G consists of T trees, then dim = #EC = T – 1.

8 Sufficient conditions for the existence of solutions

We will deal with conditions for the existence of a solution in the case of PRO-2. It is possible to derive sufficient conditions for the existence of a ranking solution for each cycle with Q constraints {'a', …'z'} in the subgraph G in K_N. Starting point is the system of equations as in eq. (3):

$$(\mu_{'a'} - \mu_{'b'}) = \gamma_{'ab'}\ sqrt(\sigma_{'a'}^2 + \sigma_{'b'}^2)$$
$$(\mu_{'b'} - \mu_{'c'}) = \gamma_{'bc'}\ sqrt(\sigma_{'b'}^2 + \sigma_{'c'}^2)$$
$$\dots$$
$$(\mu_{'z'} - \mu_{'a'}) = \gamma_{'za'}\ sqrt(\sigma_{'z'}^2 + \sigma_{'a'}^2)$$

(Q equations). In a cycle, with the constraints in a circular order, the sum of the left-hand sides equals 0. That means that the inner product of the vector Γ

$$\Gamma = (\gamma_{'ab'}, \dots, \gamma_{'za'})^t$$

with

$$(sqrt(\sigma_{'a'}^2 + \sigma_{'b'}^2), \dots, sqrt(\sigma_{'z'}^2 + \sigma_{'a'}^2))^t$$

equals zero. Solutions for the variances precisely exist if the inner product (Γ, P) attains different signs when P runs over the set of the Q *column* vectors of the following form:

1 1 0 0 0 … 0 0
0 1 1 0 0 … 0 0
0 0 1 1 0 … 0 0
0 0 0 1 1 … 0 0
…
0 0 0 0 0 … 1 1
1 0 0 0 0 … 0 1

The number of vectors in this set, as well as their dimension, equals the number of constraints.

9 The approximate case

Until now, we have dealt with the model in an analytic way, without paying attention to cases in which an exact solution does not exist, but in which an approximate solution may exist. This is especially interesting when, according to the PRO model, the dimension of the solution is less than zero. In such a case, a general exact solution does not exist, unless the probabilities P('x' < 'y') are chosen in a special way to allow the existence of a solution of an over-determined system. (This situation is comparable to the solution space of the linear equation Ax = b where rank(A) is not

maximal.) If the probabilities deviate from this special set, the exact solution disappears, but it may still be possible to define the 'best possible' solution. This 'best possible' solution minimizes the mismatch between the observed probabilities on the one hand and the predicted probabilities on the other. One appropriate approach is to define the 'best possible' model M to maximize the following likelihood:

$$P(\{P(\text{'x'} < \text{'y'} < \ldots)\}x, y, \ldots \mid M)$$

For systems with N constraints, this likelihood can be evaluated in a numerical way (the involved calculation is straightforward but tedious, it reduces an involved integral to an integral of a gaussian over a higher-dimensional octant (where all coordinates are non-negative). Figure 4 shows the situation for dimension 3. The three lines determine the boundary between 6 regions in 3 dimensional space, and the associated probability P('x' < 'y' < 'z') is the integral of a certain gaussian function over the corresponding V-shaped region. The very same principle holds in higher dimensions (but is more difficult to show in a figure).
The involved analysis shows that more complicated ranking schemes can be treated in this manner. This more complex modelling goes beyond the scope of this paper.

10 Implications for ranking in OT

In this paper, the ranking of constraints on the basis of input data, as dealt with in recent approaches in OT, was taken as a starting point. We investigated aspects of this ranking from a mathematical perspective. The results that were thus obtained can be translated back to OT in the following way.
Given a set of observed data of which the ordering of the constraints is to be modelled, it is possible to assert beforehand which combinations of pair-wise orderings can be modelled by a soft-ranking model and which combinations can not. The model that we investigated in this paper associates each constraint with a gaussian with *two* free model parameters (mean and variance). With each combination of imposed pair-wise ranking probabilities, a unique graph G has been associated. Whether this combination of probabilities can be modelled or not fully depends on the graph structure of G. This graph structure is expressed in terms of number of vertices, its number of components and faces, and the genus of G. The results, formulated as the dimension of the solution space, can eventually be summarized as: *constraints and components facilitate solutions; faces and the genus hamper solutions*.

The higher the dimension of the solution space, the more freedom exists to follow the 'recent statistics' in the input data. This means that the model is better capable to follow (i.e. explain, reproduce) fluctuations in the statistics of the incoming data, considered as a stream of input data.

For applications on the phonological domain, the results imply that the ranking of four or fewer constraints is in general straightforward, while in the case of more than four constraints the existence of an exact solution is coincidental – as a consequence, the solution is in general necessarily an approximation. The existence of (exact) solutions fully depends on the structure of the graph that is determined by the linguistic constraints.

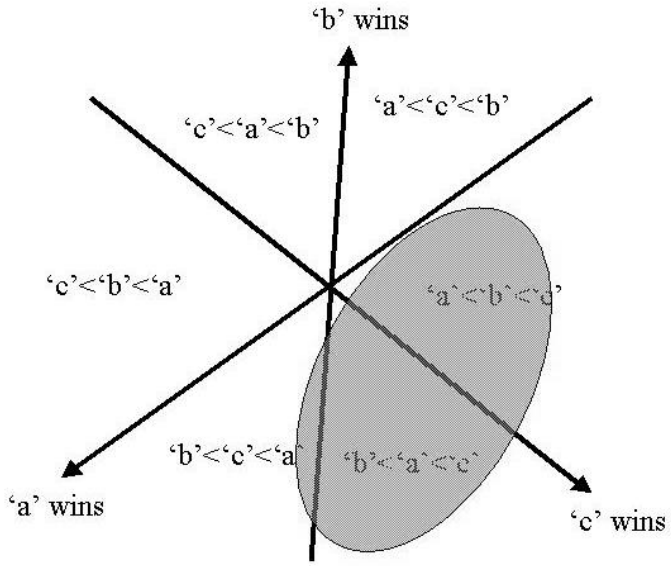

Figure 4. Evaluation of $P(\{P(\text{'x'} < \text{'y'} < \text{'z'})\}_{x,\,y,\,z} \mid M)$. This figure depicts the situation for three constraints {'a', 'b', 'c'}. For example, the probability of observing the order 'a'-'b'-'c' is given by the integral of a gaussian distribution (which is related to the model M - here represented by an ellipse) over the associated V-shaped part at the right side of the figure. The depicted model favours 'c' to be the winning constraint. Seen from this point of view, constraint ranking is a soccer game playing with the location (and shape) of the green ellipse to find an optimal location (and shape) that maximally matches all the probabilities P('x'<'y'<'z') (specified beforehand). Observe that the three vectors (indicated by ''x' wins') span a 3 – 1 = 2 dimensional subspace, and that the angle between one another is exactly 120 degrees. In the case of equal variances, the ellipse turns into a circle, and evidently the room to model the different probabilities will decrease accordingly.

Acknowledgments

We thank Paul Boersma and Louis Pols for their comments on an earlier version of this paper.

References

Ashby, F. G. (Ed.) (1992). *Multidimensional models of perception and cognition*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Ashby, F. G. & Alfonso-Reese, L. A. (1995). "Categorization as probability density estimation." *Journal of Mathematical Psychology*, **39**, 216-233.

Boersma, P. (1997). "How we learn variation, optionality, and probability." *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam*, **21**, 43-58.

Boersma, P. & Bruce H. (2001). "Empirical tests of the Gradual Learning Algorithm." *Linguistic Inquiry*, **32**, 45-86.

Charniak, E. (1993). *Statistical language learning*. MIT Press, Cambridge, Mass.

Cowel, R, Dawid, A, & Sebastiani, P. (1996). "A Comparison of Sequential Learning Methods for Incomplete Data." *Journal of Bayesian Statistics*, **5**, 581-588.
Coxeter, H.S.M. (1969). *Introduction to geometry*. John Wiley & Sons.
CYK algorithm, see http://www.cse.ucsc.edu/classes/cmps130/Fall00/CYK.html
Dempster, A.P., Laird, N.M. & Rubin, D.B. (1977). "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society B*, **39**, 1-38.
Frank, R. & Satta G. (1998). "Optimality theory and the generative complexity of constraint violability." *Computational Linguistics* **24**(2), 307-315. (Also ROA-228-1197; http://www.xrce.xerox.com/publis/mltt/pto/frank+satta1998).
Greiner, R. (1996). "PALO: A Probabilistic Hill-Climbing Algorithm." *Artificial Intelligence*, **84** (1-2), 177-208.
Haussler, D. (1990). "Probably Approximately Correct (PAC) Learning." In: *Proceedings of the 8th National Conference on Artificial Intelligence*. Boston, Massachusetts, July-August 1990, 1101-1108.
Kager, R. (1999). *Optimality Theory*. Cambridge University Press.
Kearns, M.J. & Vazirani, U.V. (1994). An introduction to computational learning theory. MIT Press.
Lee, C.H. & Gauvain, J.L. (1996). "Bayesian adaptive learning and MAP estimation of HMM." In: *Automatic Speech and Speaker Recognition* (Lee, C.H. et al, eds.) pp. 83-108.
Maddox, W. Todd & Bogdanov, S.V. (2000). "On the relation between decision rules and perceptual representation in multidimensional perceptual categorization." *Perception & Psychophysics*, **62**, 984-997.
Prince, A. & Smolensky, P. (1993). "Optimality Theory: Constraint interaction in generative grammar." *Technical Report* nr. 2, Center for Cognitive Science, Rutgers University.
Prince, A. & Smolensky, P. (1997). "Optimality: From neural networks to universal grammar." *Science*, **275**, 1604-1610.
Tesar B. & Smolensky, P. (1993). "The learnability of Optimality Theory: An algorithm and some basic complexity results." *Technical Report. Dept. of Computer Science*, University of Colorado, Boulder. (ROA-2).