Lesson 3

Understanding Data

As it will turn out, the multivariate modeling procedures are extremely mathematical, powerful, but rather mechanical procedures: There is not very much left for the user to do after the machinery has been started. But there is one thing that is common to statistics and computer programs: if you put *trash in,* you get *trash out* — and when these two, statistics and computers, are combined, as is the case in multivariate modeling, the risks only cumulate. The value of the modeling results is *completely* dependent of the quality of the modeling data; this data validity has to be ascertained by the user of the modeling tools. Whether the quality really *was* good, can only be assessed afterwards, when the constructed model is tested. It is *preprocessing of data* and *postprocessing of models* where expertise is truly needed (see Fig. 3.1).

3.1 From intuition to information

Statistical analysis methods can only do *data modeling,* not actual *system modeling.* The statistical analysis only looks for and utilizes the observed correlations between measurements. On the other hand, the mathematical tools always operate only within some selected model structure. It is the *user's* responsibility to connect this data crunching to real system modeling. When aiming at useful models, the user has to utilize his understanding in all levels of statistical modeling.

The data preprocessing and model postprocessing tasks — to be discussed later in this Lesson — are more or less *quantitative.* Before there are any numbers to be processed, however, some *qualitative* analyses are needed: The chaos of data has to structured. In this section, these preliminary analysis tasks are briefly discussed.

3.1.1 Some philosophy

It is perhaps interesting to recognize that systems modeling is closely related to those activities that have been studied by the philosophers since the dawn of history. The age-old questions of what the world is *really* like, and what we can

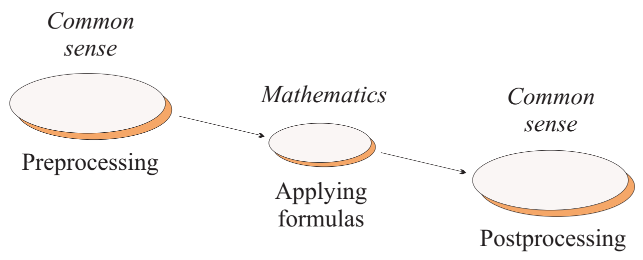

Figure 3.1: Role of knowledge in model construction

possibly know about it, are studied within two branches of philosophy, namely *ontology* and *epistemology,* respectively.

The Platonian *idealism,* where it is assumed that fundamentally there are some *ideas,* perfect objects underlying our observations, has become outdated — it has turned out that the *empiristic* approach is more fruitful. According to the Kantian view, it is assumed that it is only through our senses that we can receive information from our environment, and from these observations we construct our subjective world view, trying to find a coherent (sub-conscious) explanation for all of it. We can only hope that our mental machinery and senses are constructed so that they are capable of perceiving the essential phenomena in our environment and drawing relevant conclusions.

Loosely speaking, these two opposite views, idealistic and empiristic, correspond to the qualitative, first principles approach and the multivariate statistical approach to system modeling, respectively. It is now the mathematical machinery that is in the role of the human: Its subjective "world" is determined by those sensor signals that it is let to receive. It is not the human that is put in the center of this chaos of sensations, but it is the computer, and it is *we* that are like Gods in the universe of measurements giving the computer its senses and all those tools it has for making some sense in the chaos.

The questions of "applied ontology and epistemology" become to questions of *what are the system's real properties,* and *how can one get information about them.* What is valuable information in the measurements, and what is only noise? It is our task to make the (assumed) real system structure as visible as possible to the modeling machinery. The algorithms start from "tabula rasa", the only hardwired structures determining the construction of the data model being fixed by the organization of the measurements and the selected modeling method.

The problems of prior data analysis and manipulation, and those of model validation, are always knowledge-intensive. These tasks cannot be automated; there are just good practices that often seem to work. Because these tasks are based on expert intuition, there are no methodologies that would always work — that is why, this chapter gives various examples, hopefully visualizing the questions from comprehensible points of view.

3.1.2 Implementing structure on the data

One of the disadvantages when using data-oriented techniques is that it is difficult to integrate such models with expert understanding. However, to find the best possible model, one should utilize the available knowledge somehow. The only way to do this is to first partition the complex modeling problem into subtasks, in an engineering-like reductionistic way, exploiting the domain-area expertise in this partitioning task, and apply appropriate methods to the subproblems. If some parts of the process are known beforehand, their contribution can hopefully be eliminated from the remaining unknown behavior (see Appendix B).

To reach practical models, the data needs to be structured. This structuring should reflect the intended use of the model, but it should also support the human ways of perceiving the system.

Causality is one of the basic mechanisms that characterizes human cognition; on the other hand, statistical methods cannot see causalities (or any kinds of dependency structures) from data. This is the first, crucial task of the expert doing modeling: Determine the *inputs* and *outputs* of the system. The whole idea of regression models concentrates on modeling the relation between *action* and *reaction.*

The determination of the causal structure must be done by a human having some "common sense" — mathematical machinery can analyze data, revealing co-occurrences, but these dependencies are *correlation,* not *causation.* Constructing a causal structure that is based on false premises can result in a useless (and fallacious) model[1]. Study the following example:

> It has been recognized that *taller* children outperform smaller ones in almost all tasks, not only in physical contests, but also in cognitive tests. And this observation is *true,* however unjust it may sound. The correlation, however, vanishes, if only children of the *same age* are studied! There is no causation between size and mental capacity; rather, there is causal relation from age to both child size *and* capacity.

In concrete terms, to achieve causal structuring among data, the roles of different variables in the data vector v need to be studied. From now on, we assume that the input variables are denoted x_i, where $1 \leq i \leq n$, n being the number of input variables. The measurements are assumed to be linearly independent, so that none of the variables can be expressed as a weighted sum of the other ones. An input measurement sample can be presented as a data vector

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}. \tag{3.1}$$

Correspondingly, the output variables y_j, where $1 \geq j \geq m$, are collected in the

[1]However, this is again very much dependent of the intended use of the model: Non-causal correlations can be useful if aiming at simple prediction models, but they are not suited for control design purposes

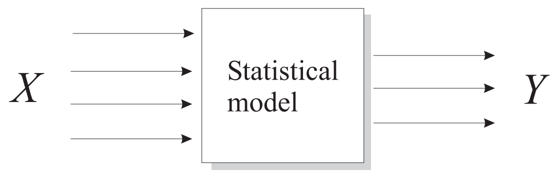

Figure 3.2: The assumed system outlook

output vector

$$y = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix}. \tag{3.2}$$

The assumption is that the process output variables can be calculated as $y = f(x)$, where $f(\cdot)$ is a linear, vector-valued function, so that $f : \mathcal{R}^n \to \mathcal{R}^m$ (see Fig. 3.2). The linearity assumption (motivated in the previous chapter) means that the mapping from input to output can be represented using the matrix formulation:

$$y = F^T \cdot x, \tag{3.3}$$

where the dimension of F is $n \times m$. Further, assume that one has measured sets of k data vectors, from $x(1)$ to $x(k)$ and from $y(1)$ to $y(k)$, respectively. These observations are written in matrices (following `Matlab` practices) as

$$\underset{k\times n}{X} = \begin{pmatrix} x^T(1) \\ \vdots \\ x^T(k) \end{pmatrix} \quad \text{and} \quad \underset{k\times m}{Y} = \begin{pmatrix} y^T(1) \\ \vdots \\ y^T(k) \end{pmatrix}. \tag{3.4}$$

Again, the mapping between these matrices can be written compactly (note that changes in ordering and the transpositions are necessary to make the dimensions match) as

$$Y = X \cdot F. \tag{3.5}$$

No structure can also be seen in the data — the structure is imposed on the data by the domain area expert. The modeling machinery will match the data against this structure, finding the best possible parameters within that framework. For pragmatic reasons, depending on the application, it can sometimes be reasonable to apply some physically non-meaningful structure for the data. For example, it can be motivated to apply a *causally incorrect* structure: If there is need for a model for estimating some quantity based on measurements of other variables, the observed correlations can be exploited regardless of the actual causality structures. The model structure should follow this intended model usage, so that the variables that are used for estimation are collected in x, and the estimated variables in y. However, it is necessary to stick to the real causality structures, if one wants to apply the models not only for prediction but also for control, etc.

3.1.3 Experiment design

Experiment design studies how maximum information can be extracted by carrying out minimum number of (expensive) experiments. If data can be measured under optimal conditions, many of the problems that will be discussed later are automatically solved. However, a more challenging case is such that one can only *observe* the system, without being able to dictate the process inputs. At least if the system is large, this assumption typically holds.

No matter whether one can carry out an explicit experimenting procedure or not, there are some necessary requirements to be taken care of before data acquisition. It needs to be noted that the algorithms only see the data that is delivered to them; in this sense, one must trust on the "benevolence of nature", so that it is not explicitly trying to fool the observer! It is, of course, quite possible that all samples happen to be located in just a narrow region of the whole operating range of the system, thus misleading the analyses, giving incorrect mean values, etc.; but it is the *law of large numbers* promising that in the long run the observed mean value, for example, should be unbiased, assuming that lots of representative data containing fresh information is collected. But this optimism is justified only if there is not some agent acting (more or less consciously) in the opposite direction, explicitly ruining the quality of data! One of such mechanisms efficiently deteriorating the quality of data is *feedback.*

In the traditional modeling, one of the most important guidelines is that *if there are some feedback loops, they should be opened* before data acquisition. Otherwise, the causality chains become blurred: The physical output, if used for feedback control, effectively becomes an input signal. Study the following example:

> Assume that the process acidity is being controlled. Because of some unmodeled disturbance, the pH value tends to fluctuate. Proportional control law is used, that is, when the acidity is too low, more acid is added, and *vice versa.* If the process acidity is analyzed statistically, it seems that *high amounts of added acid correlate with low acidity.* In this case, of course, this paradox is easily explained: The actual process dynamics being slow and noisy, it is essentially the *inverse process,* the feedback loop that is being modeled; when the process is non-acidic, the acid flow is increased. Against intuition, the "cause" now is the process acidity, the acid flow being the "effect".

However, the above view ("open all loops!") is becoming challenged in the multivariate real world. First, there are the pragmatic reasons: During on-line operation of the plant the feedbacks simply cannot be opened — and, specially when complex systems are to be analyzed, not all feedback structures can even be detected, not all dependency structures are known. And, after all, one would like to know the *typical* behaviors in the process, not the artificially induced experiments. It is the undisturbed operation of the whole plant that is actually of interest — one should model the working plant with appropriate feedbacks closed. Indeed, the system should be seen as a "pancausal" network, where all variables are tightly interconnected. The consequences of this new kind of thinking are studied closer in Chapter 11.

3.2 Selection of variables

It is assumed that all relevant information that is assumed to contribute in the model construction can be stored in the *sample vectors.* No matter what is the origin of that data, it is, after all, static mappings among sample variables in $v(\kappa)$ that are constructed. One sample, or one unit of information, is assumed to be isolated from the other pieces of information, with no memory whatsoever. Model construction tries to combine such (contradictory) pieces of information $v(\kappa)$ for different values of κ to reach a representation that can cover them all *as well as possible* — what this means is studied in detail in later chapters.

3.2.1 Feature extraction

As a basic rule, all information that is relevant for a model must be available at the same time, in a single vector $v(\kappa)$. All process phenomena should be captured as a set of stati(sti)c quantities. What is more, this data has to fulfill the structural assumptions, like linearity. Here, some guidelines are presented: How to select variables so that they would characterize the system appropriately. These variables are not necessarily the measurements directly: They are functions of the measurements that represent *features* characterizing the system appropriately. There are no unambiguous variable combinations to choose. Fortunately, the more sophisticated regression methods to be presented after Chapter 4 will efficiently solve this problem of high dimensionality, and then we can say that it is clever strategy to *include all available information there exists* and different kinds of features characterizing the model in the beginning (the excessive variables can be pruned later).

As an example, study how nonlinearities can be avoided by (formally) introducing approriate features.

Often it is so that a nonlinear function can be modeled in a linear form when the input dimension is augmented — that is, when additional features are included. This is the idea beyond, for example, *basis functions* (see later). For example, if one knows the functional form of the nonlinearity, this nonlinearity can be included among the input data: If there holds $y_i = F^T f(x)$, where f is the known classa of nonlinearity, it is possible to introduce the new input vector

$$x' = \begin{pmatrix} x \\ \hline f(x) \end{pmatrix}. \tag{3.6}$$

However, when doing data-based modeling, such *a priori* knowledge often cannot be assumed to exist. A generic way to extend the linear framework is to apply some kind of parameterized family of prototypical nonlinearities: For example, it is known that smooth functions can be approximated by their *Taylor expansions* that consist of a power series. Unknown nonlinearity forms can also be approximated using truncated power series expansions. If the linear term of variable x_i does not suffice, arbitrary number of higher-order terms can be included among the data:

$$\begin{pmatrix} x_i \\ x_i^2 \\ \vdots \\ x_i^\xi \end{pmatrix}. \tag{3.7}$$

When the nonlinear prototypical features are included among the input data, it is the task of the modeling machinery to select among the relevant components and determine their weights. More complex multivariate nonlinearities can be handled in the similar manner, applying the multiple-variable Taylor expansion, so that if one wants to be prepared for quadratic dependencies among variables x_i and x_j, the input data vector can be augmented by the following three variables: x_i^2, x_j^2, and $x_i x_j$.

3.2.2 Special challenge: Dynamic systems

In systems engineering applications, it is often the dynamic properties that are of special interest. However, if dynamic phenomena are to be modeled, one is facing a problem: Static, instantaneous features are not enough to capture the dynamics that is characterized by memory, or inertia, coupling variables together also along the time axis. The basic trick is to use time series data, that is, prior variable values, $x_i(\kappa)$, $x_i(\kappa-1)$, etc. all have separate entries in the data vector — this is the standard approch, for example, in system identification, where the dynamic system also needs to be expressed in a static form (see Sec. 10.4). System theory assures that if the system memory extends back n time steps, behaviors of a n'th order dynamic system can be captured. However, this definition of data vectors means that successive samples are overlapping, there are copies of the same variable values in different samples; this overlap and redundancy among samples can cause numerical problems (see Lesson 10).

If the dynamic system is infinite dimensional, sometimes the data representation can be simplified: For example, if there are delays, etc., the data can first be synchronized.

The above time series approach can be applied for capturing the fast dynamics in the system. However, it is not necessarily these high-frequency phenomena that are always of special interest; sometimes it is the stationary behaviors rather than the immediate transients of more or less random signal realizations that should be captured to reach some statistical relevance of features. The emphasis can be put on different frequency ranges, for example, by low-pass/band-pass filtering of the signals. Filtering of signals affects the weighting among frequency bands; this idea can be extended when one closer studies how information is not only distributed among frequencies but buried in the observations.

Signal smoothing is still not the only alternative to enhance the data: When filtering, it may be that relevant information is inevitably lost. Statistically relevant phenomena can be captured, for example, by matching the signals against some basis function families. The frequency-domain studies can be motivated also in this framework: If the signals are matched against the orthogonal set of harmonic functions, one receives spectra corresponding to the frequency content in the signals. These spectral components can be used for characterizing the dynamic properties of the system, and the data vector x can be constructed correspondingly. Whereas the spectra represent long-term phenomena, other function families can be applied to capture short-term ones: For example, *wavelets* have been proposed for this purpose. Wavelets also span a family of orthogonal functions, but having a rather limited range, they can be applied to characterize spurious peaks in signals, etc.

Powerful features can be constructed when observations are matched against assumed model structures, nonlinear or dynamic, and when it is the fitted model parameters that are used to characterize the system. For example, in cellular phones voice coding applies this strategy: The formants characterizing the voiced phonemes can efficiently be captured in the auto-regressive (AR) model parameters, and when only the dynamic model parameters are employed the amount of transmitted data can be radically reduced. The overall system behavior can be represented as a collection of local behaviors, as characterized by lower-level local model parameters within some structural framework. To implement more complicated feature extraction strategies, different approaches can be further combined: For example, time-domain (time series) structures can be combined with temporally local feature extraction techniques for modeling variability of behavioral properties between time windows.

In short, when defining features to characterize dynamic phenomena, one should avoid trusting some individual phenomena, absolute time points, etc., and use some invariant quantities or perhaps statistical cumulants characterizing signal properties. The features should be valid for different sets of signals with different noise realizations: No minimum/maximum values, etc., but averages or probabilities. Integral-based criteria (ISE, ITSE, etc.) are typically smoother-behaving than some perhaps visually well-motivated criteria[2].

Sometimes it is possible to abstract the time axis away altogether, consentrating exclusively on the higher-level *quality measures* directly [**??**]. The higher-level measures one extracts from the data, the farther the quantities are from the original measurements, and the nearer they are *qualities* characterizing the system. There is also no clear distinction between the “quantifying variables” and the “qualifying variables” — also this structure is not determined by the system itself, but by the model designer.

There are some intuitions offered also by the studies concerning *cybernetic systems:* The key point in such systems is balance, and the cybernetic model essentially is a model over the spectrum of balances. To capture this essence, one has to code these balances already in the data; that is, the tensions keeping the system in balance need to be represented by the data. In concrete terms, this means that not only the process state but also the balancing forces, or control signals, need to be included in data.

[2]For example, the *settling time,* traditional measure characterizing system responses, revealing when the oscillation after a transient has decayed below the level of, say, 5% of the original, turns out to behave in a curious, non-continuous manner: It is either after the first oscillation cycle, or after the second (or third, ...), when the criterion level is no more crossed — it turns out that as system parameters are varied the possible locations of these time points are not continuously distributed but more or less clustered along the time axis

3.3 Data preprocessing

After the set of variables has been selected, they need to be conditioned to reveal the information they carry in an optimal way. The most typical tasks here are *centering* and *scaling.* First, however, more challenging situations are studied.

3.3.1 Reaching "well-behavedness" of data

Often, the distribution of the variables is more or less peculiar. Sometimes it can be motivated to normalize the distribution — remember that it was assumed that data being modeled is Gaussian.

Qualitative data

Qualitative data here means data that does not have continuous distribution: For example, there can be binary data concerning process operating mode, etc. A single status bit can have crucial effects on the interpretations: The roles of the variables can change altogether depending of the operating mode. In principle, such qualitative data gives rise to clustering, so that each combination of qualitative variables defines a cluster of its own. However, the number of clusters explodes exponentially if there exist various qualitative variables, and this should not be done without closer analysis of data. As was discussed in Sec. 2.4.3, introducing new clusters too hastily may weaken the overall information content that is available for modeling the individual clusters. Further, there are the problems of mastering the "model library": There is a separate model for each cluster.

Assume that one allocates a separate (binary) variable for each of the qualitative values. Very often it turns out that the effects of the individual binary data become abstracted away, so that they sum up to a more or less continuous distributions.

There are different types of qualitative variables, not all are binary. Some qualitative variables can rather naturally be quantified: For example, alternatives along a continuum (like "hot" — "medium" — "cold") can be *fuzzified* by a domain-area expert, so that those variables can be coded in numeric form after all (and, as will be seen also when discussing *neural networks,* the "modern" methods should not be seen as alternatives of statistical methods, but as complementary techniques) .

Logistic regression

Sometimes one has variables that are limited to a certain range. For example, assume that the variable p_i (being interpreted as some "probability") ranges originally between 0 and 1. The problem here is that linear models cannot easily be constrained to only deliver results obeying such range limitations. It turns out that by appropriately modifying the variables, such problems can be (virtually) avoided.

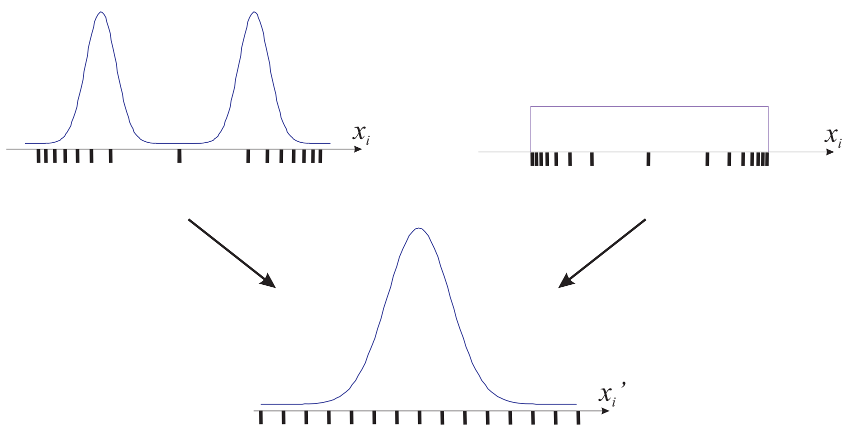

Figure 3.3: The idea of histogram deformation: The original data densities (on top) can be changed by stretching and contracting the data axis appropriately. This modification is applied to all of the variables separately

Assume that one computes $p_i/(1-p_i)$ — this way, the range can be extended from 0 to infinity. Additionally, if one defines

$$v_i = \log \frac{p_i}{1-p_i}, \tag{3.8}$$

there holds for the new variable $-\infty < v_i < \infty$. If such a data data deformation is carried out before modeling, one sometimes speaks of *logistic regression.* Note that extreme values $p_i = 0$ and $p_i = 1$ are equally illegal when applying the deformation.

Histogram equalization

In some cases there is no real physical reason to assume that the data should be non-Gaussian in the first place. It may be that the anomalies in the distribution are caused by some external factors, whereas the original distribution really was normal (see Sec. 3.6.1). In such cases one can equalize or "renormalize" the virtual distribution by nonlinear modifications of the variable scale: Data density is deformed when the samples are distributed on a differently scaled axis (see Fig. 3.3).

This deformation of the data axis must be remembered when applying the model that is constructed for renormalized data: All variables have to be deformed correspondingly. Indeed, this need for restoring the original data properties applies to all data preprocessing.

3.3.2 "Operating point"

Look at the regression formula (3.3): It is clear that the regression hyperplane always goes through the origin of the data space, so that $x = \mathbf{0}$ means $y = \mathbf{0}$. This must be taken into account during the data preprocessing: One has to select one point (explicitly or implicitly) where the regression hyperplane will be anchored. This is simple if there is some physical knowledge available: For example, if the point $\bar{\mathbf{x}}$ is known to correspond to $\bar{\mathbf{y}}$, one has to apply such a transformation that this point becomes the origin of the modified data; that is, $\mathbf{x} \leftarrow \mathbf{x} - \bar{\mathbf{x}}$ and $\mathbf{y} \leftarrow \mathbf{y} - \bar{\mathbf{y}}$. This transformation must, of course, be remembered every time when the model is used, eliminating $\bar{\mathbf{x}}$ from $\mathbf{x}_{\text{est}}$ during run-time application, and after the $\mathbf{y}_{\text{est}}$ has been found, the transformation must be inverted by adding $\bar{\mathbf{y}}$ to the result to receive the answer in original coordinates.

Often, there is no *a priori* knowledge of the values $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$. The normal approach in such cases is to *assume* that the regression line goes through the data center, that is, one selects $\bar{\mathbf{x}} = \frac{1}{k} \cdot \sum_{\kappa=1}^{k} x(\kappa)$ and $\bar{\mathbf{y}} = \frac{1}{k} \cdot \sum_{\kappa=1}^{k} y(\kappa)$, and eliminates this mean from the data. This procedure is called *mean centering* of data. Note that, even though this approach is commonly used, it is still quite heuristic[3].

In principle, there is another way to avoid this affinity problem caused by unknown operating point: One can include some constant among the measurement signals, so that, say, $x_0(\kappa) \equiv 1$; the resulting mapping $y = F^T x + F_0$ does not have the above constraints. Here this approach is not recommended, though: The problem is that the signal covariance matrix would become singular, one of the variables being constant, and some of the methods that will be discussed later could not be applied at all.

In some cases one can eliminate the effects of biases by *differentiation,* that is, one can define $x'(\kappa) = x(\kappa) - x(\kappa-1)$ and $y'(\kappa) = y(\kappa) - y(\kappa-1)$. It turns out that when using $x'(\kappa)$ and $y'(\kappa)$ rather than the original variables, the constant term vanishes from the model:

$$\begin{array}{rl} & y(\kappa) = F^T x(\kappa) + F_0 \\ - & y(\kappa-1) = F^T x(\kappa-1) + F_0 \\ \hline & y'(\kappa) = F^T x'(\kappa). \end{array} \tag{3.9}$$

It may also be so that the values $\bar{\mathbf{x}}$ and $\bar{\mathbf{y}}$ change continuously. For example, linear trends are common in practice. Elimination of the trends (or other deterministic components) is more difficult than compensating constant biases, because the behaviors rarely remain constant *ad infinitum.*

Finally, study an example: If mean centering is forgotten, and no other appropriate method is applied, the results can be catastrophic, specially if the data mean dominates over the variance: The center of the virtual distribution lies always in the origin, extending symmetrically in the "negative" direction (see Fig. 3.4). The resulting model is intuitively incorrect: If x goes up, also y goes up according to the model — even though this is evidently incorrect.

3.3.3 Data scaling

The role of variable scaling is to make the relevant features optimally visible in the data. Study an example:

[3]If the center is determined blindly from the data, the degrees of freedom become lower; when calculating covariance matrices, for instance, this should be taken into account (by dividing with $k - n$ rather than with k), but here it is assumed that the number of samples k is so large that ignoring this does not matter too much

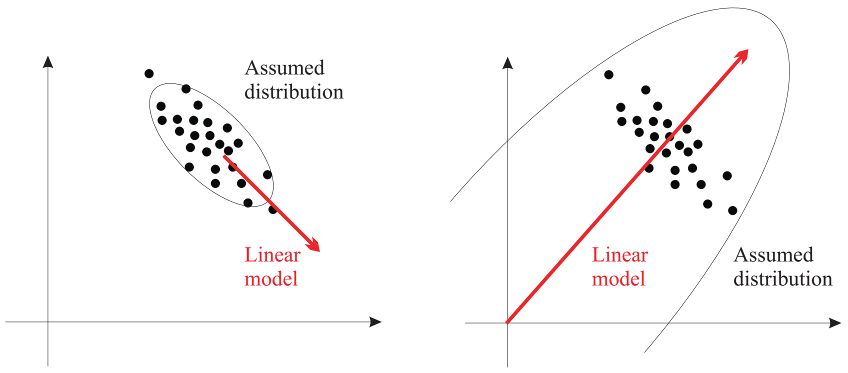

Figure 3.4: The original data distribution, on the left, and the virtual distribution if the centering is "forgotten", on the right

> Assume that there are temperature values in the range from 100°C to 200°C and pressure values in the range from 100000 Pa to 200000 Pa among the measurement data. The variation range in temperature (ignoring units) is 100 and in pressure it is 100000. It turns out that in the mathematical analysis the role of the temperature will be neglected because the variation range is so narrow: The error-square criterion concentrates on minimizing the more significant variations, emphasizing the pressure measurements exclusively.

The "equalization" of variable visibility can be carried out using data scaling. Scaling can be formally expressed using weighting matrices $W_{\mathbf{X}}$ and $W_{\mathbf{Y}}$: If data $\mathbf{X}$ is to be scaled, for example, one has $X = \mathbf{X}W_{\mathbf{X}}$. Often, $W_{\mathbf{X}}$ and $W_{\mathbf{Y}}$ are diagonal matrices with the corresponding elementwise scaling factors on the diagonal.

It is customary to assume (if there is no additional knowledge) that data given in different units should carry the same level of information. This heuristic means that all variables should have about the same variation range to be equally emphasized in the error-squared based algorithms (see Sec. **??**). To accomplish this normalization, the weighting matrix should be selected as

$$W_{\mathbf{X}} = \begin{pmatrix} \frac{1}{\sqrt{\mathrm{var}\{\mathbf{x}_1\}}} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{\sqrt{\mathrm{var}\{\mathbf{x}_n\}}} \end{pmatrix}, \tag{3.10}$$

and similarly for the output data. For each variable there then holds $\frac{1}{k} \cdot X_i^T X_i = 1/(k \cdot \mathrm{var}\{\mathbf{x}_i\}) \cdot \mathbf{X}_i^T \mathbf{X}_i = 1$. However, there are also other ways to define the scaling (see Appendix B), and there are no general guidelines for scaling that would always give optimal results exist.

However, it seems that if the data comes from a strictly cybernetic system [9.11], the rigid model structure proposes some guidelines. If the variables represent "deformations" in the cybernetic system as defined by the interaction between a system and its environment, variance normalization is explicitly motivated;

further, in such systems mean values are not necessarily eliminated in the preprocessing phase.

After the above steps it is now assumed that the data in X and Y are valid, and the system properties are (more or less) optimally visible in the data. In the subsequent chapters, the construction of the matrices X and Y is no more concentrated on.

3.4 Model construction and beyond

In the beginning of the chapter it was claimed that there is no room for expertise during the actual regression model construction phase. However, this is not exactly true. First, the *selection of the modeling method* is a question of what one expects there to be found in the data. Second, many of the more sophisticated methods are more or less *interactive,* so that the final model refinement (like determining the model order) is left to the user.

3.4.1 Analysis and synthesis

When using the more sophisticated methods, to be discussed in later chapters, the modeling procedure can be roughly divided in two parts, in *analysis* and in *synthesis.* In analysis, the mathematical machinery is used to reveal some kind of *latent structure* hidden among the observed data dependencies. The data being numeric, there are typically no clear-cut absolutely correct answers to the question of the underlying structure; the machinery just makes the data better comprehensible, so that the final decisions can easier be made by the user. This structure visualization is typically carried out so that the most fundamental data properties of the high-dimensional data are compressed into sequences of scalars measuring the relevance between the structural constructs. Generally, it is the model dimension selection that is left to the user.

After the analysis, in the synthesis phase, the analyzed structure is reconstructed in another form; when discussing regression models, this new structure emphasizes the mapping from input to output.

Note that in synthesis and in analysis the data preprocessing can be carried out in different ways — that is, the selection of the association matrix form studied in the previous chapter is an independent task from final mapping model construction. Generally, in analysis, when only the latent structure is searched for, there is more freedom; on the other hand, in synthesis, one has to be able to somehow "invert" the data deformations to construct the output mapping.

It turns out that optimizing the model is often rather simple (at least if the optimality criteria are selected in a sensible way). However, it also turns out that sometimes *the "best" is an enemy of "good".* It is not only the accuracy but also the *robustness* or *generalization capability* of the model that should be taken into account: How the model behaves when the data is somewhat different as compared to the training data?

3.4.2 Validating the model

After a model has been constructed, the knowledge is needed in *interpreting* the results: What has actually been carried out, does the model do what it was intended to do. In principle, the model validity should be checked using statistical significance tests, etc. However, these methods often turn out to have more theoretical than practical value and they are not studied in this context; a more pragmatic approach to model validation is taken.

It is fair to apply the same criterion for evaluating the model as was used when the model was constructed, that is, the sum-of-squared-errors criterion[4]. However, there is a catch: A good measure for checking the model robustness, applicable to all models, is to see what is the average prediction error size for *independent* data that has not been used for training. If the error matrix is defined as $E = Y_{\text{test}} - \hat{Y}_{\text{test}}$, where Y_{test} is the correct output and $\hat{Y}_{\text{test}}$ is the estimate given by the model, the following *Mean-Square Error (MSE)* measure can be applied for each output separately:

$$\frac{1}{k} \cdot E_i^T E_i = \frac{1}{k} \sum_{\kappa=1}^{k} e_i^2(\kappa) = \frac{1}{k} \sum_{\kappa=1}^{k} \left(y_{\text{test},i}(\kappa) - F_i^T x_{\text{test}}(\kappa) \right)^2 . \tag{3.11}$$

Without independent data, if only the data fit is measured, one can only speak of modeling *data.* If the model works fine for independent *validation data,* one can assume that the model also captures the actual *behaviors.* A still more challenging goal is to find a model that would represent the *system.* Whereas validation data is typically collected from the system in the same environmental conditions as the training data was, the *testing data* is collected in different conditions, in different time. If the correspondence between the model and the real system still is good, one can be satisfied — at least for some time: The properties of the systems typically change over time.

Because the properties of the models are determined in the preprocessing phase, but the model validity can be seen only afterwards, it is evident that the cycle between preprocessing and model construction becomes iterative. Indeed, it is clever to construct different types of models, using different kinds of preprocessings for different sets of input data, and compare the results. Rather than employing the computing capacity for complex parameter fitting for complex models once and for all, the repetitive approach is here preferred: The designs become more transparent and analyzable in this way. Because of this iterative nature of model design, it is important that the model construction can be carried out in an efficient way — and the toolbox of methods to be presented later all share this efficiency (linearity) goal.

[4]Remember that the selection of the validation criterion is not quite straightforward: For example, note that the (otherwise clever) information theoretic criteria like Final Prediction Error criterion (FPE), Akaike Information Criterion (AIC), or Minimum Description Length (MDL) criterion are *not* suited here. They only employ training data in the formulas, measuring the model applying *a priori* structural assumptions; robustness, being due to *unanticipated* disturbances, cannot be captured. Typically, it would be the basic least-squares method that would win

3.4.3 Cross-validation

A practical way to evaluate the model validity, at least if there is scarcity with data, is *cross-validation.* The basic idea is to leave one sample (or sequence of samples) out at a time from the data set, construct regression model using other remaining training samples, and check how well the missing sample can be predicted using this truncated model. When this procedure is repeated for all samples (or all sequences), a rather reliable view of how well the modeling method can abstract the data is found. On the other hand, large cross-validation errors may also reveal outliers that can be eliminated during the next modeling cycle.

3.5 Summary: Modeling procedures

The above discussion can be summarized as follows (note that the steps below illustrate the typical procedure, not always being implemented exactly in that way):

Constructing the model

1. Construction of the features, classification of data, search of primary Gaussian distributions, outlier detection, determination of training sets $\mathbf{X}$ and $\mathbf{Y}$, and the corresponding test sets $\mathbf{X}_{\text{test}}$ and $\mathbf{Y}_{\text{test}}$.
2. Determination of the data scaling matrices $W_{\mathbf{X}}$ and $W_{\mathbf{Y}}$ using data in $\mathbf{X}$ and $\mathbf{Y}$.
3. Preprocessing, data transformations, centering and scaling, giving $X = (\mathbf{X} - \bar{\mathbf{X}})W_{\mathbf{X}}$ and $Y = (\mathbf{Y} - \bar{\mathbf{Y}})W_{\mathbf{Y}}$, and $X_{\text{test}} = (\mathbf{X}_{\text{test}} - \bar{\mathbf{X}})W_{\mathbf{X}}$ and $Y_{\text{test}} = (\mathbf{Y}_{\text{test}} - \bar{\mathbf{Y}})W_{\mathbf{Y}}$.
4. Model structure refinement, if appropriate, giving $\theta = g_\theta(X', Y')$.
5. Model construction, giving $F = g_F(X, Y, \theta)$.
6. Model validation, comparing $X_{\text{test}}F$ against Y_{test}.

Note that in Steps 4 and 5 different preprocessing procedures may be used, so that the data X' and Y' need not be the same as X and Y. The semantics of "functions" g_θ and g_F will be concentrated on in subsequent chapters (see page 80).

Using the model

1. Construction of the features, classification of data, selection of the primary Gaussian distribution, outlier detection, giving $\mathbf{X}_{\text{est}}$.
2. Preprocessing, data transformations, centering and scaling, giving the final data $X_{\text{est}} = (\mathbf{X}_{\text{est}} - \bar{\mathbf{X}})W_{\mathbf{X}}$.
3. Model use, giving $\hat{Y}_{\text{est}} = X_{\text{est}}F$.
4. Inverse transformations, denormalization and decentering; reconstruction of the final estimate by inverting all preprocessing operations that were carried out for the output data: $\hat{\mathbf{Y}}_{\text{est}} = (\hat{Y}_{\text{est}}W_{\mathbf{Y}}^{-1}) + \bar{\mathbf{Y}}$.

3.6 Case studies

In this section two examples of modern process data preprocessing are presented. In both cases the information is acquired in the form of digital camera images. One reason for increased general interest in machine vision is the intuition: Humans looking at complex processes often can recognize valuable information — why not automate such a measuring process? However, whereas perceiving images is easy for humans, but pattern recognition is very difficult for computers. Assuming that there are, say, 512×512 pixels of raw data in these images, multivariate methods are clearly needed for *sensor fusion* — but without appropriate preprocessing of the data, the relevant information would still remain hidden.

These examples illustrate how difficult it is to give any general guidelines on how the data should be preprocessed; it is always a matter of domain area expertise. In 2005, both of these process analyses are still being carried out and further modeling is still continuing.

3.6.1 Analysis of the paper machine dry line

The first example illustrates the modern development work at a paper mill. This modeling effort is currently taking place at Stora-Enso Kaukopää plant in Imatra, Finland. The discussion here is somewhat streamlined, simplifying the problem to some extent; more detailed discussions can be found, for example, in [4].

The paper machine consists of the “wet end”, where the liquid-form pulp is processed, and the “dry end”, where the more or less solid-form paper (or cardboard) is received. The connection point between these two processing phases is the *wire,* where the pulp is spread from the *headbox.* The wire being a sparse fabric, excessive water is filtered through it, whereas the fibres remain on the wire, constituting the final paper formation. The wire runs continuously, taking the moist paper to the drying section.

The drying section consists of dozens of steam-heated cylinders; the important processes governing the final paper properties take place on the wire, but the results can today be measured only in the end of the dry end, causing a considerable delay in the control loop. It would be excellent if the properties of the final paper could be estimated already on the wire; this would make the control loops much faster. And, indeed, there seems to be room for improvement: The location on wire where the slush pulp turns from thick liquid into moist paper affects the paper formation and thus the final paper properties; in this transition region the mirror reflectance of the pulp surface turns into diffuse. This *dry line* has traditionally been utilized by the operators for more or less intuitive manual process control. Installing a camera beside the wire and determining the dry line from the digital image, one could perhaps mimic the expert actions (see Figs. 3.5 and 3.6).

Before some kind of control based on the dry line measurements can perhaps be implemented, the problem that remains is that the connection between the dry line measurements and quality properties at the dry end (mainly *final humidity*

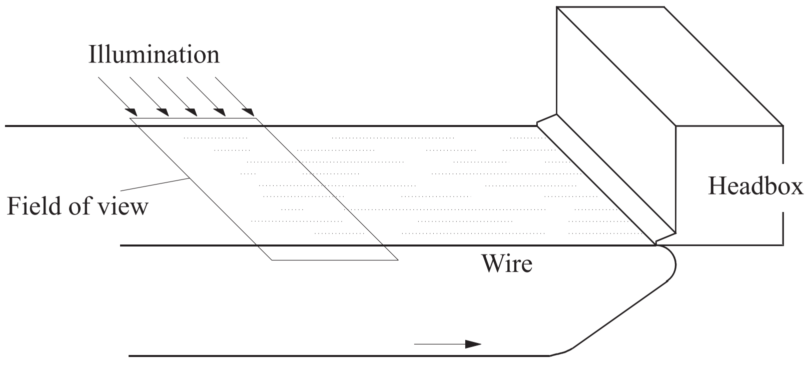

Figure 3.5: Paper machine headbox and wire

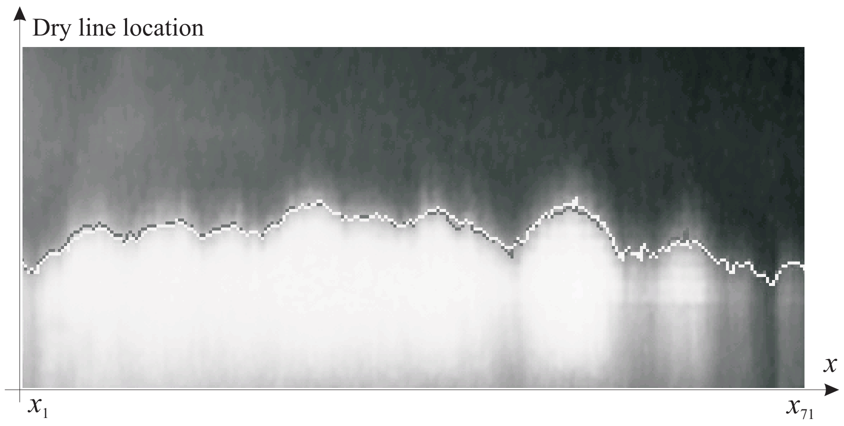

Figure 3.6: Paper machine dry line as seen by the camera. The edge detection algorithm has determined the "most probable" location of the wet/dry transition

and *total mass*) should be determined. It is the *profile* that is the most relevant now: The distribution of the fibres is determined on the wire. This means that the variations in "cross direction" (CD) in the dry and wet ends of the machine should be connected using statistical methods.

There are many technical problems in the camera imaging, concerning illumination, etc.; here we assume that these problems are solved and a high-quality digitized image is available. The pixel matrix first has to be deformed to compensate for the perspective distortions caused by the nonoptimal camera installation. An edge detection algorithm is applied to find the locations where the gradient of the pixel intensity is maximum: When these maximum gradient points are connected, an estimate for the dry line is achieved.

The dry line is analyzed every 18 seconds, extracting 71 dry line measurement values along the width of the wire, constituting the original data vector $\mathbf{x}(\kappa)$. When data is collected during several hours, the data matrix $\mathbf{X}$ can be constructed. Similarly, in the dry end, a traversing sensor measures the quality properties, constituting the output data $\mathbf{Y}$.

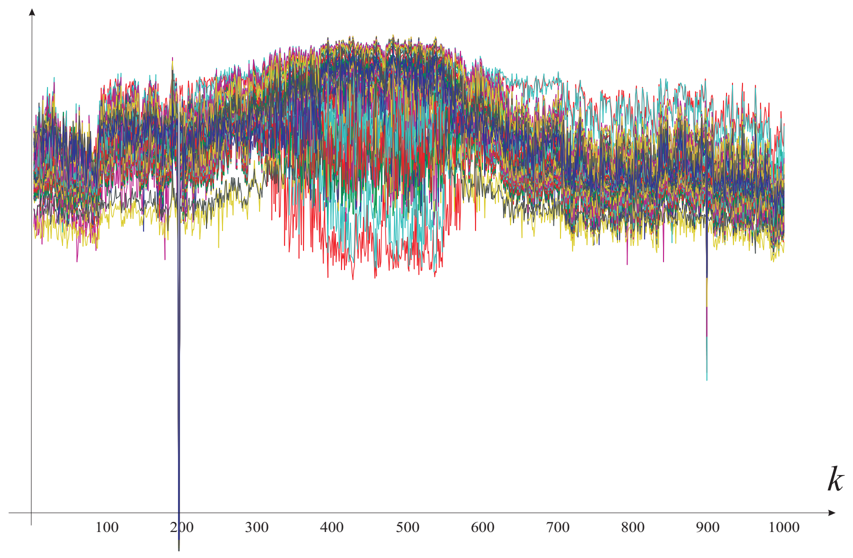

Figure 3.7: Dry line behavior in time. Some 1000 samples of all profile points (71) are plotted; samples #194 and #898 seem to be outliers

The first task in preprocessing is to make the input data and output data compatible. Because it takes (for the cardboard type being produced) 1.5 minutes to proceed from the wire to the dry end sensors, the output block must be *shifted* so that corresponding data values are aligned: The number of shifts in this case is five (because 90 sec / 18 sec = 5).

The outliers have to be eliminated from the data (see Fig. 3.7. These outliers are due to failures in pattern recognition — these problems may be caused by, for example, an operator walking by the wire, confusing the edge detection algorithm! Of course, the outlier data rows have to be eliminated in input and output blocks simultaneously to keep the data structures aligned. After this, some 1000 valid data points were left in data.

Next, the distribution of dry line points is analyzed (see Fig. 3.8). It seems that the measured dry line points seem to be distributed rather strangely, wide “unactive” regions existing between areas of frequent hits. Closer analysis reveals that this is (partly) caused by the *suction boxes:* a negative pressure under the wire is applied to increase the efficiency of pulp drainage. There is no real physical reason why the dry line should have non-Gaussian distribution, and it can be assumed that these anomalies are caused by the external suction. That is why, the normality of the distribution is restored by using histogram equalization techniques. Note that histogram equalization has to be carried out for the data still in absolute coordinates (because the effects of the suction boxes, of course, are always visible in the same coordinates), so that necessarily equalization has to precede any other manipulation affecting the numerical measurement values.

Because only profile shapes are now of interest, the changes in the dry line average have to be eliminated. This means that the instantaneous dry line mean (in cross-direction) is eliminated from the measurement sample: $\mathbf{x}'_i(\kappa) =$

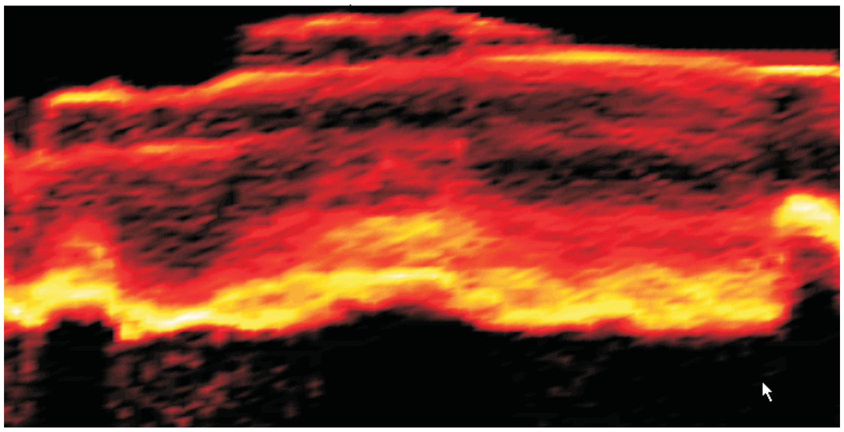

Figure 3.8: Paper machine dry line histograms as a contour plot (wire running upwards). The lighter a point is in this image, the more hits there are, meaning that the measured dry line is most probably located there in the long run

$\mathbf{x}_i(\kappa) - \frac{1}{71} \cdot \sum_{i=1}^{71} \mathbf{x}_i(\kappa)$. However, this seemingly straightforward data modification procedure introduces surprising additional problems: Note that after this modification the variables become linearly dependent, the last variable (or any of the variables) being negative of the sum of the other ones, so that, for example, $\mathbf{x}'_{71}(\kappa) = -\sum_{i=1}^{70} \mathbf{x}'_i(\kappa)$ — otherwise they would not add to zero! The easiest way to circumvent this new problem is to simply *ignore* the last, redundant variable, so that effectively we have $n = 70$. The linear independence of the variables (or the invertibility of the covariance matrix) was assumed before, and it is the prerequisite for many of the regression methods.

Only after these steps, the standard operations of mean centering and scaling are carried out (now in MD, or "machine direction", that is, κ running from 1 to 1000). Now, because all input variables have the same interpretation, it is natural to simply normalize the variances to unity before the model construction phase.

3.6.2 Modeling of flotation froth

Flotation is used in mineral processing industries for separation of grains of valuable minerals from those of side minerals. In the continuous flow flotation cell (see Fig. 3.9), air is pumped into a suspension of ore and water, and the desired mineral tends to adhere to air bubbles and rises to the froth layer where the concentrate floats over the edge of the cell; the main part of other minerals remains in the slurry. The separation of minerals requires that the desired mineral is water-repellent: In zinc flotation, this can be reached by conditioning chemicals like copper sulphate $CuSO_4$.

Flotation is one of the most difficult and challenging processes in mineral processing industry. This characteristic of the process mainly arises from the inherently chaotic nature of the underlying microscopic phenomena; there are no

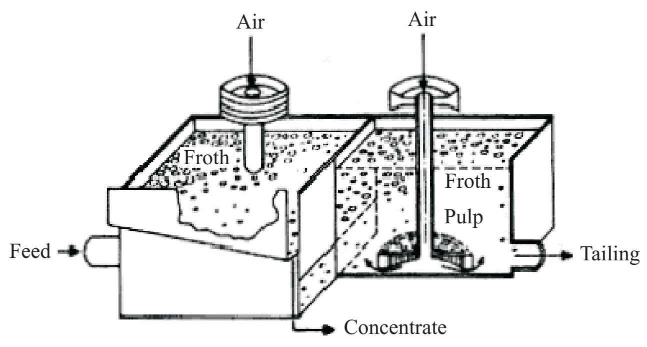

Figure 3.9: An array of two flotation cells in series

good models available that would capture the behaviour of the particles. Additional problems are caused by the fact that todays measurement technology is not able to provide with a description of the current state of the process that would be accurate and reliable enough. It is the froth surface that dictates the quality of the outflowing concentrate; the properties of the froth are reflected in its texture, movement, and colour. No standard measurement devices, however, can capture the outlook of the froth. Thus, most of the chemical reagents that are used to increase the efficiency of flotation are controlled by the human operators. The operators usually determine the suitable levels of the reagents by analysing the visual appearance of the froth; the control strategies they apply are expert knowledge.

Perhaps the limited capacity of the operator to monitor cells continuously (the operator is usually responsible for various circuits consisting of several cells) could be increased by machine vision (see Fig. 3.10)? This idea was studied during the Esprit long term research project ChaCo (Characterization of Flotation Froth Structure and Colour by Machine Vision); for example, see [25]. There were various lines in this research project; however, in this context only those studies are explained where the machine vision system was used to help the human operators in their task, analyzing the froths for process monitoring and supervision purposes.

The status of the flotation froth cannot be uniquely characterized; there are just a few measurements that can be explicitly defined and determined (like the froth level and thickness), whereas most of the factors that characterize the froth properties are more or less conceptual having no explicit definition. To construct "soft sensors" for the unmeasurable quantities, operator interviews were first carried out to find out what are the most relevant phenomena to be studied in the froth. It turned out that the trivial features — like "big bubbles", "small bubbles", etc. — are only used by novices, not the real experts. The operator interviews revealed that the most interesting (more or less fuzzy) froth types in zinc flotation are the following[5]:

[5]Note that *classification* can be seen as a special case of regression. Each of the classes has an output of its own in the regression model; this variable has value "1" if the sample belongs to that class. Using regression, the calassification results are not binary — the output values reveal how certain the classifications are

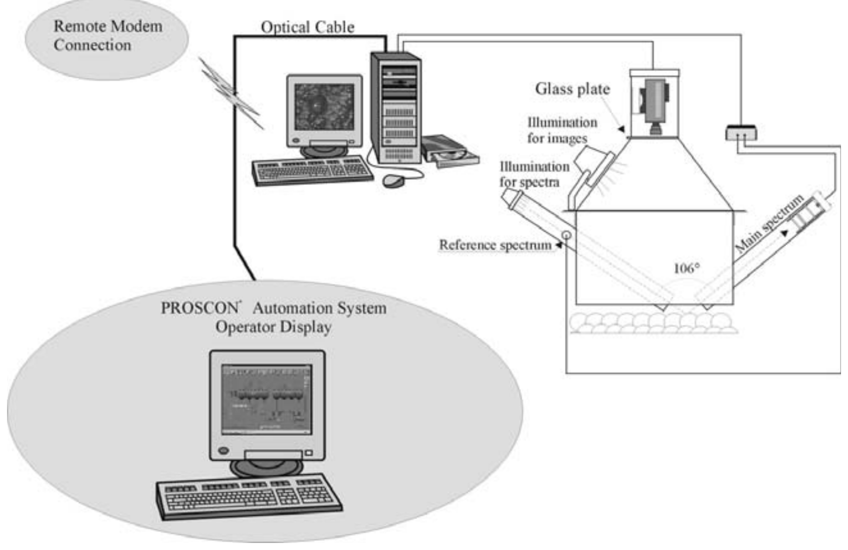

Figure 3.10: The automatic froth monitoring system helping the operator

1. **"Wet" froth** is characterized by "empty bubbles", meaning that not all bubbles are covered with the concentrate; it also seems that most of the bubbles are tiny whereas some of them may grow excessively. The bubbles do have a rather high tendency to burst.

2. **"Dry" froth** has a rather even tessellation, bubbles being of equal size and all being covered by concentrate; because of the uniformity, the bubbles are often hexagonal. This seems to be the category characterizing optimal production conditions, both froth speed and quality of concentrate being high.

3. **"Stiff" froth** is "porridge-like", the bubble forms becoming distorted and finally being substituted for layered concentrate rafts; this kind of froth floats rather unevenly, sometimes stopping altogether. In the extreme, stiffness can make the froth collapse, so that no concentrate floats out; these pathological cases should be avoided at any cost.

These characterizations are conceptual and there are no exact mathematical definitions for them (see Fig. 3.11). The role of image data preprocessing is to somehow make the classes distinguishable. First, it was noticed that static images are not enough: Many of the characterizations involve dynamic phenomena. Second, there is need for both frequency-domain and spatial segmentation approaches, as well as for pixel-wise analyses:

- **Dynamic phenomena** were captured by analysing image pairs having 0.2 sec time interval; this way, the changes between the images revealed information about the *bubble collapse rate* (how well the aligned images match each other) and *froth speed* (the average speed being determined as the maximum point in the image pair cross-correlation matrix — this can be calculated in the *frequency space,* that is, the tgwo-dimensional

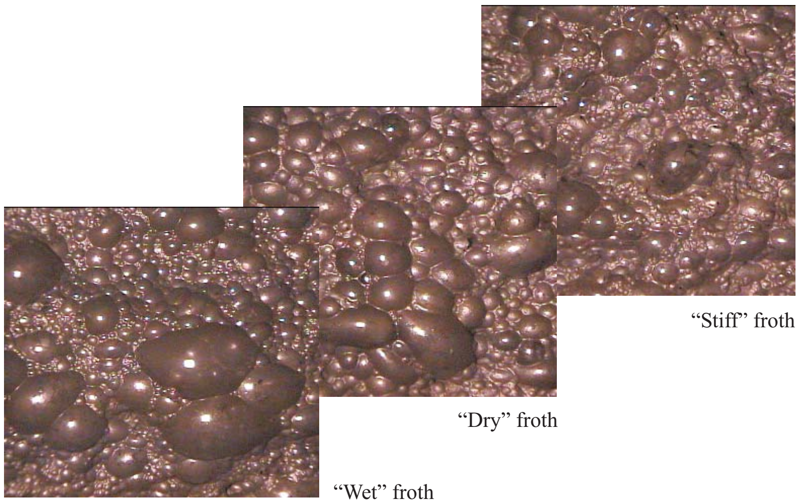

Figure 3.11: Different "conceptual categories" as seen by experts

FFT transform is applied to both images, and these transformed images are multiplied pixelwise, revealing the matches among shifted images). The *variation in speed* was measured by how high the average maximum cross-correlation peak was as compared to neighboring values.

- **Frequency domain methods** (in practice, based on the two-dimensional fast Fourier transform) were used to analyze the directional orientedness and non-sphericity of the bubbles; also the above cross-correlation matrices were calculated using FFT.
- **Segmentation techniques** (based on the so called "watershed technique") were used to extract the properties of individual bubbles; the bubble size distributions were determined this way, as well as average "roundness" of the bubbles.
- **Pixel-wise analyses** were carried out, for example, to determine the "emptiness" or transparency of the bubbles. The bubbles covered with concentrate only reflect light in a diffuse manner, whereas uncovered bubbles typically have bright total reflectance points on top; the number of maximum intensity pixels in the image can be used as a measure for the number of empty bubbles.

Finally, there are some few dozen variables characterizing the froth state, a new set of variables being calculated after every twenty seconds; this data is then mean-centered and normalized.

After all the above steps the data is ready for further model building — whatever that happens to mean. These questions will be concentrated on in subsequent chapters.

Computer exercises

1. You can check how deformed, non-Gaussian data looks like after the "equalization" of the histogram:

```
DATA = regrDataClust(1,2,100,5,3);
hist(DATA);     % Matlab histogram command
defmatrix = regrForm(DATA);
X = regrDeForm(DATA,defmatrix);
hist(X,10);
hist(X,30);     % Note the changed resolution!
```

2. Load data by running the `m`-file `dataEmotion`. There are five different signal sources (or, actually, five different "modes" of the same source!) collected in the columns of the matrix `DATA`. To have some intuition into the signals, you can try, for example

```
sound(DATA(:,1),16384);
```

   You are now asked to search for such features that these signal sources could be distinguished from each other. First divide the signals in shorter sequences of, say, a few hundred samples each, and analyze these sequences separately. Are there some invariances between sequences coming from the same source as compared to the other sources?

   In addition to the time-domain features, you should experiment with, for example, frequency domain features (use `fft`), AR-type (auto-regressive) model parameters, etc. — and you can combine these, defining new features that are based on how fast the "first-order" features change. How robust do you think your features are — would the same phenomena be detected in other samples from the same sources?