

# Multiple Model Estimation: A New Formulation for Predictive Learning

Vladimir Cherkassky and Yunqian Ma

Department of Electrical and Computer Engineering

University of Minnesota

Minneapolis, MN 55455

{cherkass, myq}@ece.umn.edu

## Abstract

*This paper presents a new formulation for predictive learning called multiple model estimation. Existing learning methodologies are based on traditional formulations such as classification or regression which assume that available (training) data is generated by a single (unknown) model. The deviations from this model are treated as zero-mean i.i.d. noise. These assumptions about underlying statistical model for data generation are somewhat relaxed in robust statistics, where a small number of outliers are allowed in the training data. However, the goal of learning (under robust statistical formulations) remains the same, i.e., estimating a single model consistent with the majority of ‘representative’ training data. In many real-life applications it is natural to assume multiple models underlying an unknown system under investigation. In such cases, training data is generated by different (unknown) statistical models. Hence, the goal of learning is to simultaneously solve two problems, i.e. to estimate several statistical models AND to partition available (training) data into several subsets (one subset for each underlying model). This paper presents generic mathematical formulation for multiple model estimation. We also discuss several application settings where proposed multiple model estimation is more appropriate than traditional single-model formulations.*

## 1 Introduction and Motivation

Various applications in engineering, statistics, computer science, health sciences and social sciences are concerned with estimating ‘good’ predictive models from available data. In such problems the goal is to estimate unknown dependency (or model) from historical data (or training data), in order to use this model for predicting future samples (or test data). The process of estimating predictive models from data involves two distinct steps [Cherkassky and Mulier, 1998; Dowdy and Wearden, 1991]:

- (1) *Problem specification*, i.e. mapping application requirements onto a standard statistical formulation. This step mainly reflects common sense and application-domain knowledge, so it cannot be formalized.
- (2) *Statistical inference*, i.e. applying constructive learning methodologies to estimate a model specified in (1) from available training data.

Even though most research is focused on theory and constructive methods for statistical inference (aka learning theory and learning methods), there exists a strong connection between the problem specification and statistical inference, as explained next. Classical statistical paradigm is based on (parametric) density estimation; under this approach any data mining application can be (at least conceptually) reduced to density estimation (from available data). However, constructive methods based on density estimation formulation do not work well with finite high-dimensional data. Recent empirical methods developed in statistics and neural networks effectively implement the idea of minimization of (penalized) empirical risk, rather than density estimation. A solid theoretical framework for such methods is provided by Vapnik-Chervonenkis (VC) learning theory [Vapnik, 1995]. Conceptually, VC-theory makes a clear

distinction between the problem specification (formulation) and solution approaches (inductive principles) – this distinction is often obscured in classical statistics. VC-theory describes 3 distinct formulations for the learning problem, i.e., density estimation, regression (estimation of continuous-valued function) and classification (estimation of indicator-valued function or class-decision boundaries). Note that VC-theory makes a strong argument that for finite sample estimation problems one should always use the most appropriate direct formulation (i.e. classification or regression) rather than general density estimation formulation [Vapnik, 1995; Cherkassky and Mulier, 1998]. Moreover, recent success of Support Vector Machine (SVM) methods (developed in VC-theory) for various data mining applications provides an empirical justification of this claim [Schoelkopf et al, 1999]. Herein lies the connection between the problem formulation and constructive learning methods for statistical inference: available learning methods are designed for specific problem formulations, but the problem formulation itself has to be meaningful, i.e. should reflect application requirements.

The major challenge for data mining applications is to come up with realistic learning problem formulations that faithfully reflect real-life application requirements [Cherkassky, 2001]. Currently, most applications are mapped onto either ‘pure’ classification or ‘pure’ regression formulation. It can be argued that such standard learning formulations do not adequately reflect many application requirements [Cherkassky, 2001]. In such cases, inadequacies of standard formulations are compensated by various preprocessing techniques and/or heuristic modifications of a learning algorithm (for classification or regression). Effectively, these practical heuristics introduce a priori knowledge on the level of constructive learning algorithm. In contrast, [Cherkassky, 2001] proposed to develop more flexible problem formulations – this can be potentially beneficial for many applications where ‘standard’ classification or regression is not

appropriate. The main assumption underlying existing statistical formulations is the assumption that all available data is generated by a single (unknown) statistical model. We call it single model estimation. Generic learning system for single model estimation is shown in Fig.1 - this system describes such common tasks as density estimation, regression and classification [Vapnik, 1995; Cherkassky and Mulier, 1998]. We briefly review next standard (single-model) regression formulation, in order to contrast it to multiple-model regression formulation introduced later.

**Standard regression formulation:** given finite number of samples called training data  $(\mathbf{x}_i, y_i), (i = 1, \dots, n)$ . Statistical model assumes that response values (output) is the sum of unknown (target) function and random error,

$$y = t(\mathbf{x}) + \delta \quad (1)$$

where  $\delta$  is i.i.d. zero mean random error (noise) and  $t(\mathbf{x})$  is a target function (ground truth). The goal of learning (estimation) is to select the best model (function)  $f(\mathbf{x}, \omega^*)$  from a set of admissible models parameterized by (generalized) parameter vector  $\omega : f(\mathbf{x}, \omega), \omega \in \Omega$ . The best model provides most accurate approximation of (unknown) target function. Various learning methods developed in statistics, neural networks and learning theory (such as MARS, CART, MLP networks and SVM) differ in the parameterization of admissible models (2), strategies for minimizing empirical risk and in methods for model selection (complexity control) – see [Cherkassky and Mulier, 1998] for details.

**Multiple model regression formulation:** Generic system for multiple model estimation is shown in Fig.2 – it describes a learning system for estimating two models from a single data set. We discuss the two-model case for the sake of explanation, and its generalization to multiple models is straightforward. In order to understand multiple model formulation in Fig. 2, it may be

instructive to focus on its differences with respect to traditional single-model estimation shown in Fig.1. The main distinction between these formulations is that under multiple model formulation data samples originate from several (generating or hidden) models (i.e. from System 1 and System 2 as in Fig.2). However, the Learning Machine has no knowledge about proper assignment of training /test data to respective models, i.e. training sample  $(\mathbf{x}, y)$  provided to the Learning Machine has no label for the underlying (generating) model as shown in Fig.2. The goal of learning now becomes two-fold, i.e. accurate estimation of each generating model AND accurate classification/assignment of the training and test data to respective models. Specific loss functions used to quantify generalization (prediction) performance of a learning method combine these two goals (as detailed later in this paper).

In the next section we describe several application domains that can be naturally described using multiple-model formulation. Section 3 provides detailed mathematical description of multiple-model formulation. Summary and conclusions are given in Section 4.

## **2 Application Examples**

This section describes several applications domains that can be described using multiple model estimation framework. We emphasize that multiple model estimation approach naturally reflects many application settings. Hence the focus is on presenting application requirements and mapping them onto a meaningful learning problem formulation, rather than describing constructive solution approaches (learning algorithms) for these problems.

*Application 1: financial engineering.* Here the goal is to develop consistently profitable ('safe') daily trading strategies for broad market indices (such as S&P500 or Dow Jones Industrial Average) or large diversified mutual funds. The motivation is based on the notion that

it may be possible to benefit from a short-term (daily) market volatility using statistically-based trading strategies. With recent computerization of the financial industry, there is a growing trend towards negligible (zero) trading cost; as witnessed by a rapidly growing segment of the mutual funds industry (Rydex, Potomac and ProFunds) catering specifically for short-term traders [WSJ, 1999]. For such mutual funds, we consider daily trading strategies that make a decision (whether to stay invested in the market or switch into cash) in the end of each trading day, based on current market indicators. Clearly, if we think that tomorrow the market (or given index/mutual fund) will move UP, we should stay fully invested for possible appreciation; alternatively, if we think the market will move DOWN, we should switch into cash for capital preservation (safety). Hence, optimal trading strategies achieve statistically optimal trade-off between preservation of principal and short-term (daily) gains. Application of statistical learning methodologies to estimating optimal trading strategies can be described as follows [Cherkassky et al, 2000]. Formally, trading strategies can be represented as a set of indicator (BUY or SELL) decision functions  $f(\mathbf{x}, \omega)$  depending on a set of (proprietary) market indicators (at the end of each trading day) denoted by  $x$  and parameterized by parameters  $\omega$ . An optimal parameter vector resulting in optimal trading strategy  $f(\mathbf{x}, \omega^*)$  is estimated using available (historical) data. This problem statement effectively follows standard single-model formulation, i.e., the goal is to estimate a single model (trading strategy) from available data. Arguably, this objective itself may be unattainable, since it is well-known that the market conditions and market psychology changes very fast, in response to unanticipated economic and political events. Consequently, a single-model estimation approach results in strategies that are too complex and dangerously non-robust. Alternatively, the problem of estimating robust trading strategies can be formulated under *multiple-model* framework, by noting that the stock market undergoes 3 medium-term market

conditions, i.e. *medium-term trend* is UP, DOWN or FLAT. The medium-term trend conditions can be detected using standard methods in technical analysis, such as plotting moving averages etc. Further, it makes perfect sense to estimate optimal short-term (daily) trading strategies for a given (known) medium-term trend. For example, if the medium-term trend is UP, a good short-term strategy would be buying on the dips; if the trend is DOWN, a good short-term strategy would be selling on the rallies. Using multiple-model formulation, the goal of learning is to partition available training data into 3 data sets corresponding to different market conditions (UP, DOWN and FLAT market) and to estimate 3 daily trading strategies optimized for each market condition. We also note here that in this application the number of distinct models is known (given a priori). We also emphasize that under multiple-model learning, partitioning of available data into several subsets and model estimation (for each subset of data) are not two independent problems. That is, we *can not* approach its solution by first partitioning (clustering) available data into several subsets, and second using single model estimation techniques to model the data in each subset. Constructive solution approaches for multiple model estimation will be discussed later in a companion paper.

*Application 2: motion analysis in computer vision.* Let us consider the problem of motion estimation from a time sequence of 2-dimensional image data  $\mathbf{x}(t)$ , where  $\mathbf{x}(t)$  denotes two-dimensional coordinates of a point (pixel) in an image at discrete time  $t$  (present time). Coordinates of this point at the next time moment  $t+1$  can be represented as a function of its coordinate at present moment  $t$ :

$$\mathbf{x}(t+1) = \mathbf{x}(t) + W\mathbf{x}(t) + W_0 \quad (2a)$$

where the motion parameters are encoded in matrices  $W$  and  $W_0$  as

$$W = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix} \quad \text{and} \quad W_0 = \begin{pmatrix} w_{01} \\ w_{02} \end{pmatrix}$$

That is, parameters in  $W_0$  represent translations in the horizontal and vertical dimensions, whereas parameters in  $W$  represent affine motions (such as rotation, shear etc.). Motion type and motion parameters are usually unknown and need to be estimated from available data. In a simple case when all image pixels follow the same (unknown) motion, the problem of motion analysis leads to a standard (linear) regression formulation:

$$\mathbf{y}(t) = \mathbf{x}(t) + W\mathbf{x}(t) + W_0 + \delta \quad (2b)$$

where the response vector  $\mathbf{y}(t) = \mathbf{x}(t+1)$  for notational convenience, and an additive noise  $\delta$  is included to reflect the effects of measurement noise. Representation (2a) and (2b) are equivalent, since both put in correspondence similar image locations, drawn from two consecutive image frames drawn at time  $t$  and  $t+1$ , respectively. However, representation (2b) can be readily interpreted using standard regression formulation (1), so that one can model (estimate) the dependency of ‘response’ variables  $\mathbf{y}(t)$  on the input variables  $\mathbf{x}(t)$  using finite amount of available data. Hence standard learning/estimation techniques can be readily applied to the problem of motion estimation. The practical objective of such motion estimation may be making accurate predictions for image pixels not included in the training set (‘interpolation’) and/or making predictions for future frames (‘extrapolation’). The main issue in motion analysis is determining correct motion type and estimating its parameters from finite available (training) data. This involves statistical issues of model selection and complexity control – see, for example, [Wechsler et al, 2002] who successfully applied VC-based model selection to the problem of motion analysis and motion parameter estimation.

Standard regression formulation (2b) for motion analysis works perfectly well for estimating motion parameters when all image pixels follow the same (unknown) motion, i.e. rotation, pure translation etc. However, there are many practical situations where different parts of an image participate in different motions. Then motion analysis becomes more challenging: we need to partition available image data into several subsets AND (at the same time) estimate motion parameters using an appropriate subset of data. This problem (of simultaneous estimation of multiple motions) is known as *spatial partitioning* in Computer Vision (CV) literature. Specifically, we are interesting in *non-disjoint* spatial partitioning settings where parts of an image experience *non-disjoint* motions, i.e. the trajectories of image pixels involved in different motions may overlap in the input space. This setting can be contrasted to the case when different motions occupy disjoint regions of the input space (known as *disjoint spatial partitioning* in CV), where one can apply well-known partitioning techniques for standard regression formulation (i.e., regression trees or mixture of experts) that partition the input ( $\mathbf{x}$ ) space and estimate a different model (motion) in each disjoint input region. For the problem of multiple motion estimation (assuming non-disjoint case), we are given training data  $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_n, \mathbf{y}_n)$ , however it is not known which portion of an image (which samples) are involved in each motion. This problem can be naturally formulated using multiple-model estimation framework (see Fig.2), so that each model represents an unknown motion, and the goal of learning is to separate available data into several subsets and to estimate regression models (motions) describing each subset. The number of different motions (models) is generally unknown; however the parametric form of different possible motions is usually *known* from the Computer Vision literature. For example, motion models are described by linear parametric models that include 2D linear affine, simplified quadratic flow etc. [Black et al, 1997; Shashua and Wexler, 2001]. In general, the problem of

(multiple) motion analysis is considered a challenging problem in CV. Even though many learning algorithms have been proposed, notably suitable modification of methods from robust statistics [Black et al, 1996; Chen et al, 2000], there is a general feeling that existing learning heuristics have limited applicability. For example, the method described in [Chen et al, 2000] appears to work well for 2D data; however the authors note that ‘its practicality for high dimensions is not obvious’. We may argue that the main conceptual problem with existing learning methods for multiple motion estimation in CV is that they all assume standard single-model regression formulation. Instead, one should use multiple-model formulation that is more appropriate for this application.

*Application 3: home insurance fraud detection.* Large insurance companies often employ data-driven models for identifying fraudulent home insurance claims. The goal is to identify ‘bad risk’ customers in order to drop their insurance coverage and thus avoid future losses. Currently, fraudulent claims are identified by detecting abnormal frequency and/or claim amount of insurance claims. For example, if a homeowner submits 3 or more claims over a 3-year period, then his/her home insurance coverage may not be renewed. Using such fixed-rule-based criteria for identifying fraudulent claims does not always make good business sense, because they do not adapt to random unpredictable events (i.e., natural disasters). Therefore, a better approach is to develop flexible data-driven strategies for identifying abnormal claims based on available (historical) data of past claims. This approach assumes single statistical model which relates a response variable (some quantifiable measure of the claim amount and frequency) to several input (predictor) variables comprising the property value, length of past insurance coverage, geographical location of the property (i.e., the state, rural vs metropolitan area location) etc. Assuming that such a stable dependency exists, it can be estimated from available historical data

using well-known regression estimation techniques. Then for a given test input (a set of input variables), observing large deviations from this regression model can be used as an indicator of ‘abnormal’ claims and serve as a basis for dropping insurance coverage. The learning problem (as stated above) assumes a single underlying model, i.e. a *single dependency* of the ‘normal’ claim amount/frequency on a set of input variables. It may be reasonable to describe this application using a *multiple-model formulation* approach, assuming that available data is generated by three underlying models, that is normal claim dependency for expensive homes, normal claim dependency for medium-priced homes and normal claim dependency for inexpensive homes. Under this approach, the goal of learning/estimation is simultaneous partitioning of available data into three subsets and estimating regression model for each subset of data. Note that the problem of data partitioning cannot be solved by simply splitting the data based on the estimated dollar value of home prices. This is because the concept of ‘expensive’ or ‘cheap’ home depends strongly on the geographical location and other factors (input variables).

As evident from the above application examples, using multiple-model formulation relies heavily on a priori (application-domain) knowledge. Good understanding of an application domain is important for two reasons. First, the choice between single-model formulation and multiple-model formulation cannot be made on the basis of theoretical analysis or some ‘good’ analytical properties of a particular formulation – this choice simply reflects common sense and sound engineering as applied to the problem specification. Second, multiple-model formulation is inherently more complex than single-problem formulation (because we need to estimate *several models* from a single data set). Consequently, constructive learning algorithms for multiple-model formulation need to incorporate more a priori knowledge (compared to algorithms for single-model estimation) in order to make estimation/learning tractable. In particular, such a

priori knowledge may include the number of (hidden) models (i.e., three medium-term market conditions describing stock market behavior) and specific parametric form of each model (i.e., translation and rotation motions in the motion analysis application).

### 3 Formulation for Multiple Model Estimation

This section presents mathematical formulation for multiple model estimation, which can be relevant for describing many real-life problems (i.e. example applications presented in Section 2). The description focuses on the issues important for distinction between (proposed) multiple-model formulation and traditional single-model formulation. In addition, we discuss the connection and differences between multiple-model formulation and several well-known partitioning learning methods (such as tree partitioning and mixture of experts approach) originally developed for single model formulation.

*Statistical model for data generation:* assumes that response (output) values are generated by several (unknown) models,

$$y = t_m(\mathbf{x}) + \delta_m \quad \mathbf{x} \in X_m \quad (3)$$

where  $\delta_m$  is i.i.d. zero mean random error (noise), and (unknown) models are represented by target functions  $t_m(\mathbf{x}), m = 1, \dots, M$ . We assume that the number of models is finite but generally unknown. In some applications, however, the number of models is given a priori. Statistical model further assumes that the input samples for model  $m$  are generated according to some distribution with (unknown) p.d.f.  $p_m(\mathbf{x})$ . The ‘prior probability’ of model  $m$  samples is denoted as  $c_m$ , where  $\sum_{m=1}^M c_m = 1$ . These prior probabilities are not known, but can be estimated from (large) training data set. We also assume additive unimodal symmetric zero-mean noise  $\delta_m$  for

each model (3). To simplify presentation, we assume that the noise distribution is the same for all models ( $\delta_m = \delta$ ) and this noise is described by unknown (but stationary) noise density  $\delta(y)$ .

Generic formulation for multiple model estimation in Fig.2 allows for two distinct settings:

1. *Piecewise-disjoint formulation.* The input ( $X$ ) domains for different generating models are disjoint, i.e.  $X_l \cap X_m = \emptyset$  if  $l \neq m$ . Each of the hidden models applies to different (disjoint) regions of the input space; however, partitioning of the input space into disjoint regions is not known (given) a priori and needs to be estimated from data. Here the goal of learning may be minimization of the traditional prediction risk (estimation accuracy) as in single model formulation, or identification of abnormal future samples. This interpretation is somewhat similar to partitioning methods for standard single model estimation (i.e., the mixture of experts (ME) or tree-based methods such as CART). Such partitioning methods estimate an unknown function (model) by partitioning the input space into several regions and estimating a simple model for each respective region, in a data-dependent fashion. The main difference between (existing) partitioning methods and (proposed) multiple model estimation setting is that partitioning methods represent a solution approach for standard single-model formulation. For example, ME assumes particular parameterization (mixture model) for single-model estimation. Hence, partitioning methods (such as ME) emphasize ‘smooth’ transitions between adjacent regions, due to overlapping nature of gaussian mixture. In contrast, multiple model estimation allows for sharp (discontinious) transitions between adjacent regions. Whereas any comparisons between multiple model setting and partitioning methods is ultimately application-dependent and will depend on application-specific loss function, it may be possible to perform empirical comparisons between partitioning methods and multiple-model approach for standard single-model piecewise function estimation (using

standard loss functions for classification or regression). For example, piecewise-linear function estimation can be interpreted as a single model estimation problem (using a particular parameterization of a complex single model) or under multiple-model formulation where the goal is to estimate several simple (linear) models appropriate for different (disjoint) regions of the input space.

2. *Non-disjoint formulation.* The input domains for different models are identical:  $X_l = X_m$ . Each of the hidden models in Fig.2 applies to the whole input space. In other words, x-distribution of data is *the same* for all models. So the difference (between models) is exclusively due to different distributions in y-space. Here the likely goal of learning may be accurate classification of future  $(\mathbf{x}, y)$  samples.

We present next several example data sets intending to illustrate the difference between single model formulation and multiple model formulation in general, and between existing partitioning methodologies (for single model estimation) and multiple-model estimation in particular. Figs.3-5 show representative toy data sets and corresponding model estimates (obtained from this training data) under different modeling assumptions (i.e. single-model vs. multiple-model approach). The data sets are intentionally simple (i.e., univariate regression setting) and are used to show (informally and intuitively) specific settings for data generation when the proposed multiple-model estimation approach makes sense. Data set in Fig.3 is an example of standard regression formulation (1); however it assumes discontinuous target function defined in two (disjoint) regions in the input space. Technically, this data set can be modeled using traditional single-function estimation methodology that would enforce continuous transition between the two regions, as shown in Fig.3a. This approach, however, may result in low model estimation accuracy especially when (unknown) target function has a large jump at

discontinuity. Alternatively, the same data set can be modeled under the proposed multiple-model approach as shown in Fig. 3b. An estimate shown in Fig. 3b consists of two separate models defined in two disjoint regions of the input space. The advantages of this multiple-model approach are two-fold: first, it provides better estimation accuracy (smaller generalization error), and second, it can yield useful model interpretation (i.e., the existence of two distinctly different models) important for many applications. Further, the data set in Fig.3 is a simple example of (more realistic) higher-dimensional data sets for which many existing piecewise modeling (or partitioning) methodologies (such as tree partitioning, mixture of experts etc.) have been proposed under standard single-model formulation. Hence, it may be interesting to perform empirical comparisons (in terms of prediction accuracy) between existing partitioning methods and estimation methods based on the proposed multiple-model approach, for such data sets (both synthetic and real-life). Data set in Fig.4 shows ‘non-standard’ regression problem where the two components of the target function partially overlap at the point of discontinuity. Under single-model estimation approach, the data in the overlapping region is interpreted as having very high noise, and hence the jump (discontinuity) will be smoothed leading to very inaccurate model in this region (see Fig. 4a). In contrast, the proposed multiple-model estimation would produce two distinct models that are partially overlapping in the input space, as shown in Fig. 4b. Clearly, this results in a more accurate model; however, the multiple-model approach here yields a totally new type of model. That is, it produces two different response values for the same input (in the overlapping region). Hence, it cannot be directly compared to standard partitioning methods developed under single-model formulation. Finally, the data set in Fig. 5 shows an extreme situation when the input domain of the two hidden models is the same. In this case, the single model approach is completely inappropriate (as shown in Fig. 5a), whereas the proposed approach enables accurate estimation of both hidden models (see Fig. 5b).

*Training/learning phase:* given finite number of samples or training data  $(\mathbf{x}_i, y_i), (i = 1, \dots, n)$ .

The objective of learning is two-fold:

(a) to estimate  $M$  target functions from a set of admissible models:

$$f_m(\mathbf{x}, \omega_m), (\omega_m \in \Omega_m, m = 1, \dots, M) \quad (4)$$

where  $\Omega_m$  is a parametric space for model  $m$ . Each model estimate approximates the corresponding target function  $f_m(\mathbf{x}, \omega_m^*) \rightarrow t_m(\mathbf{x})$ .

(b) to partition available training data into  $M$  subsets, where each training sample is assigned to a model. This may also (implicitly) partition the input ( $\mathbf{x}$ ) and/or output ( $y$ ) space into  $M$  disjoint regions.

As evident from the above description, statistical model for multiple-model estimation can be viewed as a generalization of the traditional single-model estimation (also see Figs. 3-5 for intuitive justification of the multiple-model approach). Alternatively, multiple-model estimation setting can be viewed as a combination of traditional classification and regression formulations. That is, we seek to partition a given data set (training data) into several subsets (classes), and at the same time, to estimate an accurate regression-like model for each subset of data

Next we consider the operation/prediction phase of a learning system under multiple-model formulation. Recall that under traditional (single-model) approach deduction amounts to estimating response  $\hat{y}$  given test input  $\mathbf{x}$ , simply as  $\hat{y} = f(\mathbf{x}, \omega^*)$ , where  $f(\mathbf{x}, \omega^*)$  is a model estimated from past (training) data. For multiple-model setting this approach would not work, because for a given test input we need first to choose an appropriate model, and only then estimate its response for given input. Further, we generally cannot select an appropriate model using the test input  $x$  alone, since the input domains for different models may be overlapping

(see Figs. 4 and 5). Hence, selecting correct model during the operation stage should be based on the  $(\mathbf{x}, y)$  values of the test data as shown in Fig. 6(b). After the model  $m$  is chosen, estimated response is generated as  $\hat{y} = f_m(\mathbf{x}, \omega_m^*)$ .

*Operation/prediction phase.*

For a given test sample  $z = (\mathbf{x}, y)$  generated by model  $k$  (which is unknown) and a set of models estimated during training stage:

$$f_i(\mathbf{x}, \omega_i^*), \mathbf{x} \in X_i (i = 1, \dots, M) \quad (5)$$

we need to perform the following two tasks.

*Task 1: Model assignment.* Determine ‘correct’ or most likely model for a given test sample  $z = (\mathbf{x}, y)$ . This is accomplished using (application-dependent) ‘distance’ between the test sample and each of the models (5). Here each model (5) is defined as a region in the input ( $\mathbf{x}$ ) space and the mapping  $f_m : \mathbf{x} \rightarrow y$  in this region. Hence, the ‘distance’ may be defined in the input ( $\mathbf{x}$ ) space or in the y-space, or some combination of both. Formally, this can be expressed as:

$$m = \arg \min_i \{ \beta * \text{dist}(\mathbf{x}, X_i) + (1 - \beta) * \text{dist}(y, f_i(\mathbf{x}, \omega_i^*)) \} \quad (6)$$

where parameter  $\beta \in [0, 1]$

Note that the ‘distance’ in the input space depends on the model densities  $p_m(\mathbf{x})$  and on prior probabilities in a rather complicated manner, when the model densities are overlapping (non-disjoint). However, in this paper we consider two special cases for statistical model for data generation (3) that is disjoint formulation and non-disjoint formulation. For *disjoint formulation*, it is natural to set  $\beta = 1$ , so the correct model is determined using some distance in the input space. For *non-disjoint formulation*, we assume that all models have identical densities in the input space (however, models may have different prior probabilities). In this case,  $\beta = 0$ , and a

natural notion of ‘distance’ may be the probability of misclassification. Hence, an optimal rule for selecting correct model in this case is given by:

$$m = \arg \min_i \{1 - P[M_i | (\mathbf{x}, y)]\} \quad (7)$$

where  $P[M_i | (\mathbf{x}, y)]$  is a posterior probability of selecting the model  $M_i$  given a test sample  $z = (\mathbf{x}, y)$ . This probability can be expressed in terms of the noise density in the statistical model (3), under the assumption that different models in (3) have the same noise distribution:

$$P[M_i | (\mathbf{x}, y)] = c_i \delta(y - f_i(\mathbf{x})) \quad (8)$$

*Task 2: Estimate improved response using selected model.* For multiple model setting, both the input and the response values of the test sample  $(\mathbf{x}, y)$  are given, and this sample is first assigned to an appropriate hidden model  $m$  and then an improved estimate for response  $\hat{y}$  is obtained as  $\hat{y} = f_m(\mathbf{x}, \omega_m^*)$ .

Next we provide quantifiable metrics for prediction risk/loss appropriate for multiple model estimation, with understanding that for practical problems such metrics have to be consistent with application requirements. The problem is to estimate loss/risk for given test sample  $z = (\mathbf{x}, y)$  with respect to the model selected during operation/prediction phase, where the goal is to obtain an improved estimate for response  $\hat{y}$  (as outlined above). Let us define two components of prediction loss for multiple model formulation. The first component measures *model estimation accuracy* (under the assumption that test samples are classified correctly in Task 1 above). The second component quantifies loss due to *model misclassification* (i.e. incorrect classification of test samples in Task 1). We introduce and discuss each component separately, since their relative importance may be strongly application-dependent. The *model estimation risk* is a straightforward generalization of (prediction) risk under traditional single-model formulation

[Vapnik, 1999, Cherkassky and Mulier, 1998]. That is, for each model  $i$  we measure the discrepancy between the (known) estimate  $f_i(\mathbf{x}, \omega_i^*)$  and the unknown (target) function  $t_i(\mathbf{x})$  averaged with respect to (unknown) distribution of input samples for this model:

$$R_i(\omega_i) = \int_{X_i} [t_i(\mathbf{x}) - f_i(\mathbf{x}, \omega_i^*)]^2 p_i(\mathbf{x}) d\mathbf{x} \quad (9)$$

Note that we used squared-error loss as in standard regression formulation; however one can use any reasonable loss function in expression (9) consistent with application requirements (i.e., linear loss, classification error etc.). Further we calculate total model estimation risk by averaging over all models:

$$R(\omega_1, \omega_2, \dots, \omega_M) = \sum_{i=1}^M c_i R_i(\omega_i) \quad (10)$$

Expressions (9) and (10) for prediction risk represent straightforward generalization of prediction risk for single-model estimation, under the assumption that each test sample is correctly classified, i.e. assigned to its generating model.

*The model misclassification* risk measures the cost of misclassification of test samples. Even though (arguably) one can use traditional measures such as classification error (in standard classification formulation) we introduce a new measure that is more appropriate for multiple model formulation. Recall that for the test sample  $(\mathbf{x}, y)$  that is classified as originating from hidden model  $m$  an estimated response  $\hat{y}$  is obtained as  $\hat{y} = f_m(\mathbf{x}, \omega_m^*)$ . Hence, if this sample has been misclassified and the actual (true) model is model  $k$  rather than  $m$ , the loss due to such misclassification can be quantified as  $L(\hat{y}, t_k(\mathbf{x}))$  using some loss function. For example, using squared loss, we have the following loss due to sample from model  $k$  classified as originating from model  $m$ :

$$L(\hat{y}_m, t_k(\mathbf{x})) = \left( f(\mathbf{x}, \omega_m^*) - t_k(\mathbf{x}) \right)^2 \quad (11)$$

In order to quantify the expected model misclassification risk we need to average loss (11) over unknown distributions. This is shown next under two simplifying assumptions. First, we assume only two models. Second, we assume non-disjoint formulation, where  $x$ -distribution of data is identical for all models, so a test sample  $(\mathbf{x}, y)$  is assigned to one of the two models based on the distribution in  $y$ -space. A general rule for assigning a test sample to one of the two models is based on posterior probabilities, i.e. test sample  $(\mathbf{x}, y)$  originates from Model 1 if  $\hat{P}[M_1 / (\mathbf{x}, y)] > \hat{P}[M_2 / (\mathbf{x}, y)]$ , and from Model 2 otherwise. Recall that a given test sample  $(\mathbf{x}, y)$  can be generated by one of the two models  $M_1 = t_1(\mathbf{x})$  or  $M_2 = t_2(\mathbf{x})$ . These models are unknown, but we have model estimates obtained from the training data as  $\hat{y} = f_1(\mathbf{x}, \omega_1^*)$  and  $\hat{y} = f_2(\mathbf{x}, \omega_2^*)$ . Hence, for a given test sample the model misclassification loss (for the 2-model case) when the test sample assigned to Model 1 has been actually generated by Model 2 can be expressed as:

$$L[(\mathbf{x}, y), M_1 / M_2] = \left( f_1(\mathbf{x}, \omega_1^*) - t_2(\mathbf{x}) \right)^2 * P(M_1 / M_2) \quad (12)$$

where  $P(M_1 / M_2)$  is the probability that the test sample assigned to model  $M_1$  has been generated by model  $M_2$ . Likewise, the model misclassification loss in the case when the test sample assigned to Model 2 has been actually generated by model 1 is given by:

$$L[(\mathbf{x}, y), M_2 / M_1] = \left( f_2(\mathbf{x}, \omega_2^*) - t_1(\mathbf{x}) \right)^2 * P(M_2 / M_1) \quad (13)$$

and  $P(M_2 / M_1)$  is the probability that the test sample assigned to model  $M_2$  has been actually generated by model  $M_1$ . Averaging model misclassification loss over (unknown) distribution of  $x$ -samples yields expected model misclassification risk:

$$R(M_1, M_2) = c_2 \int_X (f_1 - t_2)^2 P(M_1 / M_2) p(\mathbf{x}) d\mathbf{x} + c_1 \int_X (f_2 - t_1)^2 P(M_2 / M_1) p(\mathbf{x}) d\mathbf{x} \quad (14)$$

where we used for notational brevity,  $f_1 = f_1(\mathbf{x}, \omega_1^*)$ ,  $f_2 = f_1(\mathbf{x}, \omega_2^*)$ ,  $t_1 = t_1(\mathbf{x})$  and  $t_2 = t_2(\mathbf{x})$ .

Assuming known unimodal noise density  $\delta(y)$  in the statistical model  $y = t_m(\mathbf{x}) + \delta$  we can analytically evaluate  $P(M_1 / M_2)$  and  $P(M_2 / M_1)$  as:

$$P(M_1 / M_2) = \int_{-\infty}^{\frac{f_1+f_2-t_2}{2}} \delta(y) dy \quad \text{and} \quad P(M_2 / M_1) = \int_{\frac{f_1+f_2-t_1}{2}}^{+\infty} \delta(y) dy \quad (15)$$

Figure 7 shows probabilities (15) under the assumption that  $t_1 < t_2$ .

Now *total prediction risk* can be formed as a (weighted) sum of model estimation risk (10) and expected model misclassification risk (14). The goal of learning under multiple model formulation is to partition the data (into several models) and to estimate parameters of each model, in order to minimize total prediction risk.

#### 4 Summary and Discussion

This paper introduced new formulation for predictive learning called multiple model estimation. We have discussed several application domains where multiple model formulation can be naturally applied, and introduced mathematical problem statement for multiple-model regression.

In conclusion, we discuss general properties of constructive learning methods for multiple model estimation. Our goal here is not to focus on particular constructive learning algorithms (this is a subject of companion paper), but rather to discuss/introduce general statistical properties of such algorithms. As evident from the mathematical problem statement (given in Section 3), multiple-model estimation aims at solving simultaneously two problems: partitioning/clustering available data into several subsets (unsupervised learning), and estimating a model for each

subset (via supervised learning). Moreover, the prediction risk for multiple model formulation was introduced in Section 3 based on the supervised learning formulation. Hence, constructive learning methods for multiple model estimation are closely related to robust methods for supervised learning. For example, for multiple regression formulation (described in Section 3) the data can be modeled by several regression models. Assuming that the initial goal (of a learning method) is to estimate a single (dominant) regression model from all available data, such a learning method has to be:

- (1) Very robust, i.e. can tolerate large percentage of ‘outliers’. In the case of multiple model formulation the notion of outliers includes not only deviations (noise) with long-tailed distributions, but also structured outliers corresponding to other models aka ‘data with multiple structures’[Chen et al, 2000].
- (2) Very stable, i.e. can provide an accurate and stable estimate of the dominant model, in spite of high degree of potential variability in the data due to the presence of outliers (other models).

In conclusion, we give a brief discussion of desirable properties and issues for learning (estimation) algorithms for three distinct settings: standard regression under single-model formulation, robust regression under single-model formulation, and robust regression under multiple-model formulation:

- *Standard regression.* Here the goal is to estimate a single model using all available data. Statistical model for data generation assumes that data is formed by (unknown) single model (target function) corrupted by additive noise. Typical methods are based on least-squares minimization (under the assumption of gaussian noise). The accuracy of regression estimates improves as the sample size grows large. The main modeling issue is flexible estimation (when the parametric form of the target function is unknown) and

model complexity control, i.e. selecting optimal model complexity for given training sample;

- *Standard robust regression.* Here the goal is to estimate a single model using all available data. Statistical model assumes that data is generated by (unknown) single model corrupted by (unknown) noise. The goal is robust estimation (of a single model) when the model estimates are not (significantly) affected by (unknown) distribution of noise. The main modeling issue is obtaining robust estimates (of a single model) under many possible noise models, i.e. a mixture of gaussian noise and another noise density. Standard robust algorithms work only when the *absolute majority* of data samples (typically over 80%) are generated by a single model;
- *Robust regression for multiple-model estimation.* Here the goal is to estimate a single model using a portion of available data. Statistical model assumes that training data is generated by several (unknown) models, where each model may be corrupted by (unknown) noise. The main challenging issue for a learning algorithm is using an appropriate subset of available data for estimating a single (dominant) model. Here the robustness of a learning algorithm refers to ignoring multiple secondary structures when estimating the dominant model. Therefore, such algorithms should work well when *relative majority* of data samples are generated by a single (dominant) model.

**Acknowledgement:**

this work was supported in part by NSF grant ECS-0099906. The concept of multiple model estimation has been initially motivated by the problem of motion analysis in Computer Vision introduced to one of the authors (V. Cherkassky) by Prof. H. Wechsler from GMU.

## REFERENCES

- [1] V. Cherkassky and F. Mulier, *Learning from Data: Concept, Theory and Methods*, John Wiley & Sons, 1998
- [2] S. Dowdy and S. Wearden. *Statistics for Research*, John Wiley & Sons, New York, 1991
- [3] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, 1995
- [4] B. Scholkopf, J. Burges and A. Smola, ed., “Advances in Kernel Methods: Support Vector Machine”, MIT Press, 1999
- [5] V. Cherkassky, “New Formulations for Predictive Learning”, Plenary Lecture, ICANN 2001, Vienna, Austria, 2001
- [6] V. Cherkassky, F. Mulier and A. B. Sheng, “Funds exchange: an approach for risk and portfolio management”, in Proc. IEEE Conf. on Computational Intelligence for Financial Engineering, pp 3-7, New York, 2000
- [7] The Wall Street Journal, “Trading pace in mutual funds quickens”, Sept 24, 1999
- [8] H. Wechsler, Z. Duric, F. Li and V. Cherkassky, “Motion Prediction Using VC-Generalization Bounds”, *Proc. ICPR 2002* (To appear), 2002
- [9] A. Shashua and Y. Wexler, “Q-Warping: direct computation of quadratic reference surfaces”, *IEEE Trans. PAMI*, 23, 8, pp. 920-925, 2001
- [10] M. J. Black and P. Anandan, “The Robust estimation of multiple motions: parametric and piecewise-smooth flow fields”, *Computer and Image Understanding*, 63, 1, pp.75-104, 1996
- [11] M. J. Black, Y. Yacoob and S. X. Ju, “Recognizing human motion using parameterized models of optical flow”, in *Motion-Based Recognition*, S. Mubarak and R. Jain, Ed. Kluwer, Boston, 1999, pp. 245-269
- [12] H. Chen, P. Meer and D. Tyler, “Robust Regression for Data with Multiple Structures”, in *CVPR 2001, Proc. IEEE Computer Society Conf.* vol 1 pp 1069-1075, 2001
- [13] V. Vapnik, *The Nature of Statistical Learning Theory*, 2<sup>nd</sup> Ed. Springer, 1999
- [14] P. Meer, “Introduction, Robust Computer Vision: An interdisciplinary challenge”, *Computer Vision and Understanding* 78, pp 1-7, 2000
- [15] C. V. Stewart, Robust Parameter Estimation in Computer Vision, *SIAM Review*, Vol. 41 No. 3, 513-537, 1998

### Figure Captions

**Fig.1.** Learning system for single model estimation

**Fig.2.** Learning system for multiple model estimation

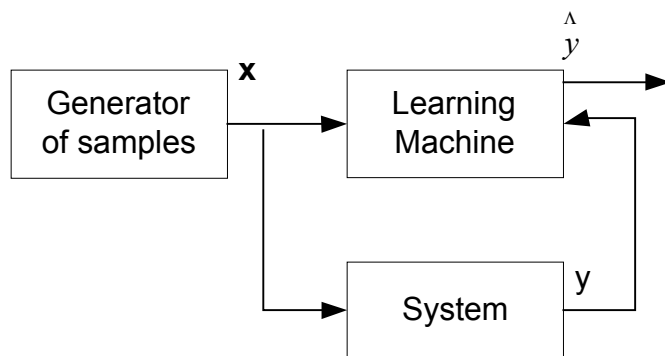
**Fig.3.** Traditional Regression setting with discontinuous target function and disjoint support in the input ( $\mathbf{x}$ ) space (a) Single model estimation (b) Multiple model estimation

**Fig.4.** Non-standard regression setting with discontinuous target function and partially overlapping support in the input ( $\mathbf{x}$ ) space (a) Single model estimation (b) Multiple model estimation

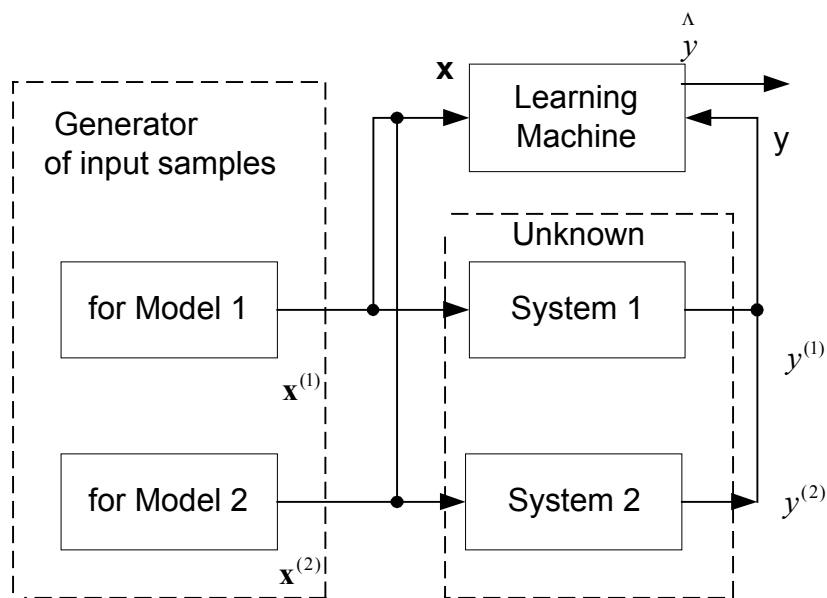
**Fig.5.** Multiple model regression setting with the same support in the input ( $\mathbf{x}$ ) space for both models (a) Single model estimation (b) Multiple model estimation

**Fig.6.** Prediction/ Operation stage of a learning system (a) Appropriate model chosen using test input ( $\mathbf{x}$ ) (b) Model chosen using  $(\mathbf{x},y)$  values of test data

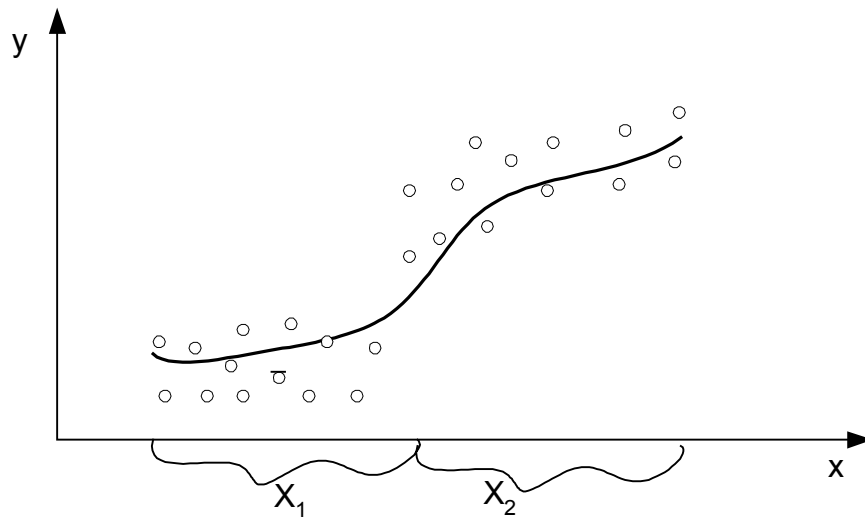
**Fig.7.** Probabilities of misclassification for the two-model case



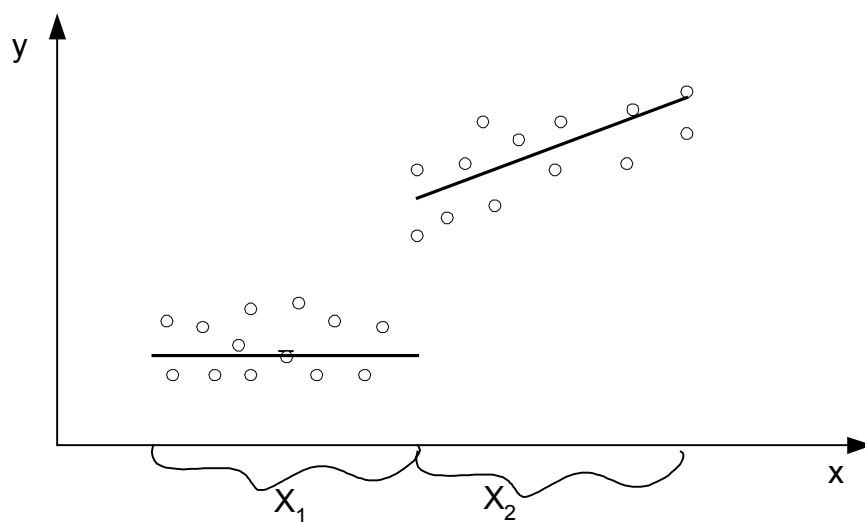
**Fig. 1.** [Cherkassky and Ma]



**Fig. 2.** [Cherkassky and Ma]

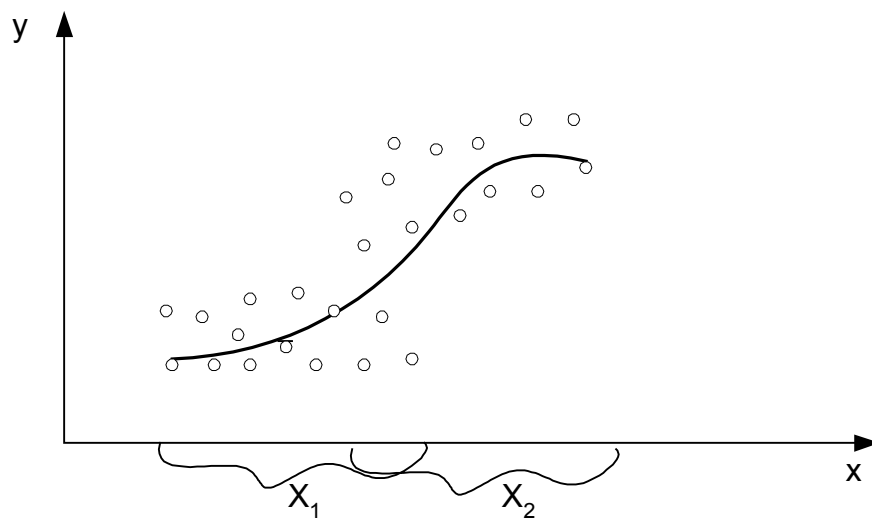


(a) Single model estimation

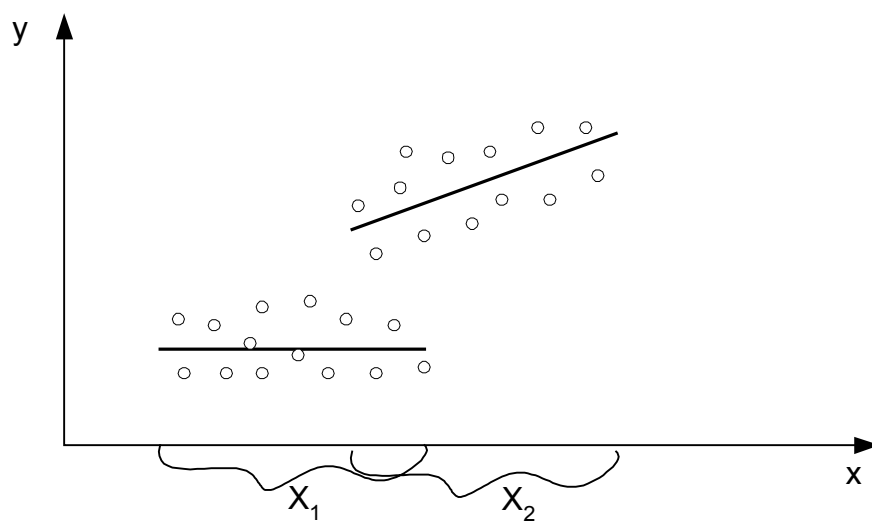


(b) Multiple model estimation

**Fig.3.** [Cherkassky and Ma]

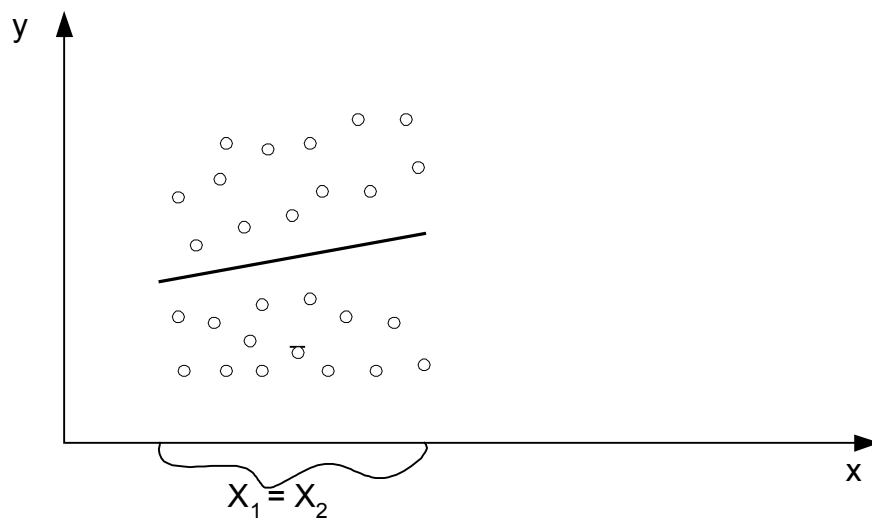


(a) Single model estimation

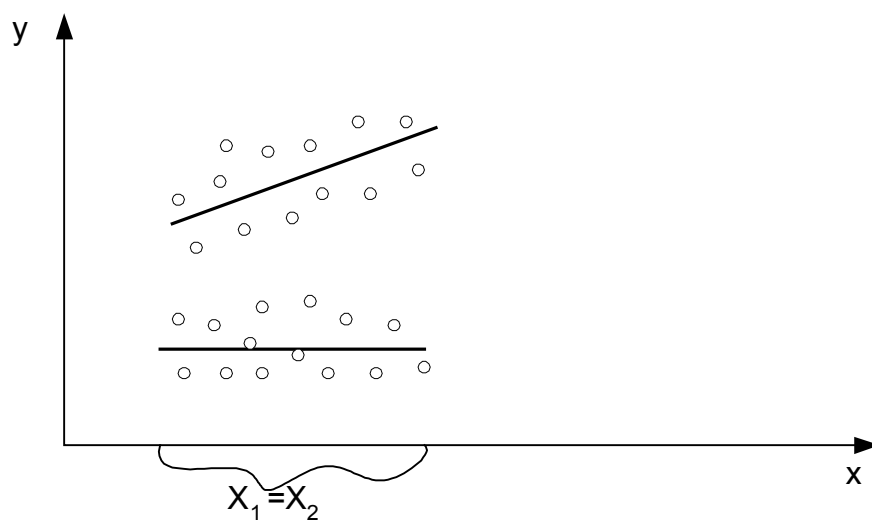


(b) Multiple model estimation

**Fig. 4.** [Cherkassky and Ma]

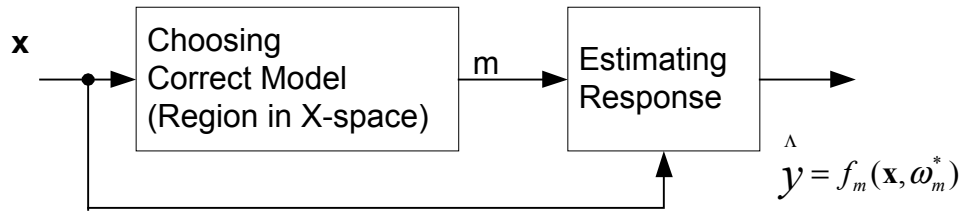


(a) Single model estimation (totally wrong)

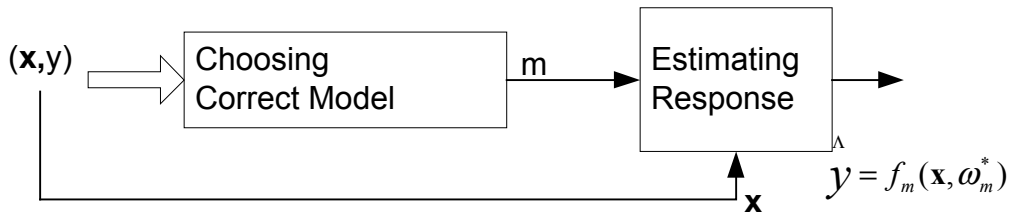


(b) Multiple model estimation

**Fig. 5.** [Cherkassky and Ma]

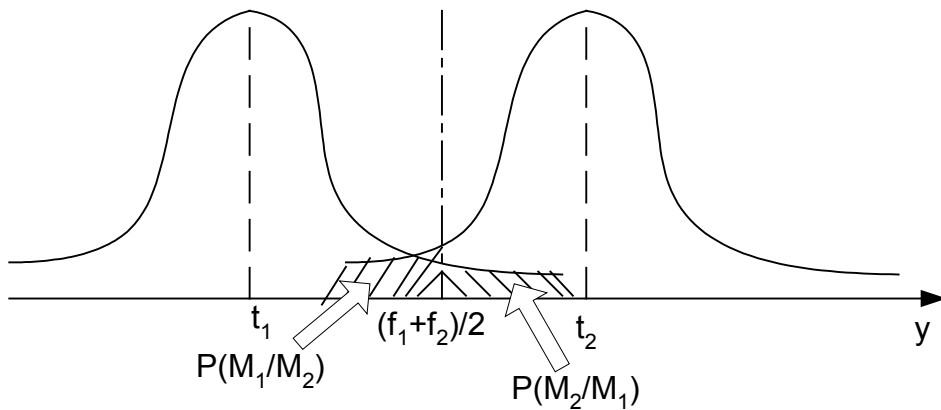


(a) Appropriate chosen using test inputs ( $x$ )



(b) Model chosen using  $(x, y)$  values of test data

**Fig.6.** [Cherkassky and Ma]



**Fig.7.** [Cherkassky and Ma]