Journal of Machine Learning Research x (2003) xx–xx Submitted 5/02; Published xx/03

Extensions to Metric-Based Model Selection

Yoshua Bengio and Nicolas Chapados
{BENGIOY,CHAPADOS}@IRO.UMONTREAL.CA
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Québec, H3C 3J7, Canada

Editor: Isabelle Guyon and André Elisseeff

Abstract

Metric-based methods have recently been introduced for model selection and regularization, often yielding very significant improvements over the alternatives tried (including cross-validation). All these methods require unlabeled data over which to compare functions and detect gross differences in behavior away from the training points. We introduce three new extensions of the metric model selection methods and apply them to feature selection. The first extension takes advantage of the particular case of time-series data in which the task involves prediction with a horizon h. The idea is to use at t the h unlabeled examples that precede t for model selection. The second extension takes advantage of the different error distributions of cross-validation and the metric methods: cross-validation tends to have a larger variance and is unbiased. A hybrid combining the two model selection methods is rarely beaten by any of the two methods. The third extension deals with the case when unlabeled data is not available at all, using an estimated input density. Experiments are described to study these extensions in the context of capacity control and feature subset selection.

Keywords: Metric-based Methods, Model Selection

1. Model Selection and Regularization

Supervised learning algorithms take a finite set of input/output training pairs $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, sampled (usually independently) from an unknown joint distribution $P(X, Y)$, and attempt to infer a function $g \in \mathcal{G}$ that minimizes the expected value of the loss $L(g(X), Y)$ (also called the *generalization error*). In many cases one faces the dilemma that if $\mathcal{G}$ is too "rich", which often occurs if the dimension of X is too large, then the average training set loss (*training error*) will be low but the expected loss may be large (*overfitting*), and vice-versa if $\mathcal{G}$ is not "rich" enough (*underfitting*).

In many cases one can define a collection of increasingly complex function classes $\mathcal{G}_0 \subset \mathcal{G}_1 \subset \cdots \subset \mathcal{G}$ (although some methods studied here work as well with a partial order). In this paper we focus on the case in which the function classes differ by the dimensionality of the input X. *Model selection* methods attempt to choose one of these function classes to avoid both overfitting and underfitting. Variable subset selection is a form of model selection in which one has to select both the number of input variables and the particular choice of these variables (but note that the two decisions may be profitably decoupled). One approach to model selection is based on *complexity penalization* (Vapnik, 1982, 1998, Rissanen, 1986, Foster and George, 1994): one first applies the learning algorithm to each of the function classes (on the whole training set), yielding a sequence of hypothesis functions $g_0, g_1, \ldots$. Then one of them is chosen based on a criterion that combines average training loss and a measure of complexity (which usually depends only on the function class $\mathcal{G}_i$). Ideally the criterion estimates or bounds generalization error (e.g. as with the Structural Risk Minimization; Vapnik, 1998). Another approach to model selection is based on *held-out data*: one estimates the generalization error by repeatedly training on a subset of the data and testing on the rest (e.g. using the bootstrap, leave-one-out or K-fold cross-validation). One uses this generalization error estimator in order to choose the function class that appears to yield the lowest generalization error. Cross-validation estimators have been found to be very robust and difficult to beat (they estimate generalization error almost unbiasedly, but possibly with large variance). The *metric-based* methods introduced by Schuurmans (1997) and Schuurmans and Southey (2002) are somewhat in-between in that they take advantage of data not used for training in order to introduce a complexity penalty or a regularization term. These methods take advantage of *unlabeled* data: the behavior of functions corresponding to different choices of complexity are compared on the training data and on the unlabeled data, and differences in behavior that would indicate overfitting are exploited to perform model selection, regularization, or feature selection.

Regularization methods are a generalization of complexity penalization methods in which one looks for a function in the larger class $\mathcal{G}$ that minimizes a training criterion that combines both the average training loss and a penalization for complexity (e.g. minimizing curvature Poggio and Girosi, 1990). These approaches are not restricted to a discrete set of complexity classes and may potentially strike a better compromise between fitting the training data and avoiding overfitting.

An overview of advances in model selection and feature selection methods can be found in a recent Machine Learning special issue (Bengio and Schuurmans, 2002), and of course in the other papers of this special issue. This paper proposes new model selection methods and applies them to feature selection. The application to feature selection is based on forward stepwise selection to select the feature subsets (among features subsets of a given size) prior to applying the model selection (to select the number of features, i.e. the size of the subsets), as explained in section 3.3.1 and in Algorithm 1, where the model selection is assumed to be performed through a functional S that takes a set of learning algorithms $(A_0, A_1, \ldots, A_n)$ along with a data set D as arguments, and returns an integer index $i^* \in \{0, \ldots, n\}$ representing the selected model. In this setting, a "learning algorithm" is represented by a functional A that takes a data set D and returns a learned function. For example A could correspond to minimizing an empirical error over a fixed class of functions.

This paper proposes extensions of previously-proposed metric-based model selection methods and applies them to feature selection problems. In section 3 we present a **first extension** that takes advantage of the particular case of time-series data in which the task involves prediction over a given horizon. The idea is to use at t the unlabeled data that just precedes t but cannot be used for training. Experimental results on feature selection for auto-regressive data are reported in section 3.3. For time-series data, a common feature selection problem is that of choosing the appropriate *lags* of the input time-series. Even though the series is one-dimensional, time-series prediction is a high-dimensional learning problem when there are long-term dependencies and one considers all the past values as potential inputs.

Algorithm 1 Stepwise Feature Subset Selection by Model Selection

Input: data set D, learning algorithm A, model selection algorithm S, loss functional L, and set of input features F. Write $V(A, F_N)$ for the restriction of A that uses only the inputs features in the subset of N input features F_N. Write $L(A, D)$ for the empirical loss (training error) when training A on D, and write $i^* = S((A_0, A_1, \ldots, A_n), D)$ the output of a model selection algorithm, which returns the *index* of one of the algorithms in $(A_0, A_1, \ldots, A_n)$ based on data D.

```
• Let $n = |F|$ the number of input variables.
• Let $F_0 = \{\}$ an initially empty set of features.
• Let $A_0 = V(A, F_0)$ the algorithm that does not use any input.
• For $N = 1$ to $n$
        • Let $f^* = \text{argmin}_{f \in F \setminus F_{N-1}} L(V(A, F_{N-1} \cup \{f\}), D)$
        • Let $F_N = F_{N-1} \cup \{f^*\}$
        • Let $A_N = V(A, F_N)$
• Let $i^* = S((A_0, A_1, \ldots, A_n), D)$
```

Output: the selected subset F_{i^*} and the selected model A_{i^*}.

The **second extension**, presented in section 4, takes advantage of the different error distributions between cross-validation and other model selection methods: cross-validation tends to have a larger variance and is unbiased. The idea is to combine the functions selected by both methods. This "meta" model selection method is rarely beaten by any of the two methods, as shown in the results of section 4.1, also with feature selection for auto-regressive data. The **third extension** (section 5) allows to apply metric methods when unlabeled data is not available at all. To obtain unlabeled data, one simply samples from an estimated input density (trained on the input part of the training data). Model selection experiments are performed on both artificial and real data sets using a Parzen windows to estimate the input density, and showing results often as good as when unlabeled data is available.

2. Metric-Based Model Selection and Regularization

Metric-based methods for model selection and regularization are based on the idea that solutions that overfit are likely to behave very differently on the training points and on other points sampled from the input density $P_X(x)$. This occurs because the learning algorithm tries to reduce the loss at the training points (but not necessarily elsewhere since no data is available there), whereas we want the solution to work well not only on the training points but in general where $P_X(x)$ is not small. These metric-based methods are all based on the definition of a *metric* (or pseudo-metric) on the space of functions, which allows to judge how far two functions are from each other:

$$d(f, g) \overset{\text{def}}{=} \psi(E[L(f(X), g(X))])$$

where the expectation $E[\cdot]$ is over $P_X(x)$ and ψ is a normalization function. For example with the quadratic loss $L(u, v) = (u - v)^2$, the proper normalization function is $\psi(z) = z^{1/2}$. Although $P_X(x)$ is unknown, Schuurmans (1997) proposed to estimate $d(f, g)$ using an average $d_U(f,g)$ computed an *unlabeled set* U (i.e. points x_i sampled from $P_X(x)$ but for which no associated y_i is given). In what follows we shall use $d_U(f,g)$ to denote the distance estimated on the unlabeled set U:

$$d_U(f,g) \overset{\text{def}}{=} \psi\left(\frac{1}{|U|}\sum_{i\in U} L(f(x_i), g(x_i))\right) \tag{1}$$

The metric-based methods proposed in Schuurmans (1997), Schuurmans and Southey (2002) are based on comparing $d_U(f,g)$ with the corresponding average distance measured on the *training set* T:

$$d_T(f,g) \overset{\text{def}}{=} \psi\left(\frac{1}{|T|}\sum_{i\in T} L(f(x_i), g(x_i))\right) \tag{2}$$

In addition, the following notation is introduced to compare a function to the "truth" (the conditional distribution $P_{Y|X}$):

$$d(f, P_{Y|X}) \overset{\text{def}}{=} \psi(E[L(f(X), Y)]) \tag{3}$$

and on a data set S:

$$d_S(f, P_{Y|X}) \overset{\text{def}}{=} \psi\left(\frac{1}{|S|}\sum_{i\in S} L(f(x_i), y_i)\right). \tag{4}$$

Schuurmans (1997) first introduced the idea of a metric-based model selection by taking advantage of possible violations of the *triangle inequality*, with the **TRI model selection algorithm**, which chooses the last hypothesis function f_l (in a sequence $f_0, f_1, \ldots$ of increasingly complex functions) that satisfies the *tri*angle inequality $d_U(f_k, f_l) \le d_T(f_k, P_{Y|X}) + d_T(f_l, P_{Y|X})$ with every preceding hypothesis f_k, $0 \le k < l$. Improved results were described in Schuurmans and Southey (2002) with a new penalization model selection method, based on similar ideas, called **ADJ**, which chooses the hypothesis function f_l which minimizes the *adj*usted loss

$$\hat{d}(f_l, P_{Y|X}) \overset{\text{def}}{=} d_T(f_l, P_{Y|X}) \max_{k<l} \frac{d_U(f_k, f_l)}{d_T(f_k, f_l)},$$

See Schuurmans and Southey (2002) for more detailed justification and for experiments showing that these two methods outperform classical model selection procedures (including cross-validation) on some small artificial data sets (with between 10 and 30 training examples) on which overfitting can be severe.

This idea of a metric-based method to control overfitting was extended to the regularization paradigm with the **ADA** algorithm, which in the case of regression is defined by the following regularized training criterion:

$$\min_{f\in\mathcal{F}} d_T(f, P_{Y|X}) \max\left(\frac{d_U(f,\phi)}{d_T(f,\phi)}, \frac{d_T(f,\phi)}{d_U(f,\phi)}\right)$$

where ϕ is a kind of prior function which can be chosen for example as the constant average of the y_i's. Even better results were obtained with this method, and comparisons were performed Schuurmans and Southey (2002) that show ADA not only to beat in most cases a wide array of model selection and regularization techniques, but also to often beat the oracle model selection method (that picks f_i that minimizes $d_U(f_i, P_{Y|X})$ out of a finite set). This may occur because ADA is a regularization technique that can thus make a finer trade-off between overfitting and underfitting. Note however that our experiments suggest that ADA involves an optimization problem which can sometimes be tricky and requires significantly more computation than the other metric model selection methods.

It is interesting to point out here that the metric model selection methods have something very important in common with Vicinal Risk Minimization (VRM) (Chapelle et al., 2001). Both seem to work by penalizing changes in behavior of $f(x)$ in the vicinity of x, using an estimation of $P(x)$ around x. VRM can be framed as the minimization of a convolved loss, i.e. expecting that $L(x, y)$ is minimized not only at the training example (x_i, y_i) but also at x's in the neighborhood of x_i, according to an input density model.

3. Extension to Time-Series Forecasting

We now turn to the case of applying statistical learning algorithms to time-series data, such as economic or financial data, that may be non-stationary. At time t, it is possible to obtain an information set $\mathcal{I}_t$ which includes all measurable observations at time t and prior to time t. It is desired to forecast some aspect $y_{t+h} = y(\mathcal{I}_{t+h})$ of this information set at a future time $t + h$, using some aspect of the information available at t, $x_t = y(\mathcal{I}_t)$. Because of the possible non-stationarity of the data (dependence on t of $P_t(y_{t+h}|x_t)$), estimating generalization error is often done with the **sequential validation** technique, rather than with leave-one-out or K-fold cross-validation. Sequential validation is based on the analysis of the sequence of losses obtained by sliding a learning algorithm A over the time sequence, as shown in Algorithm 2.

Algorithm 2 Sequential Validation

Input: data sequences $\{x_t\}, \{y_t\}$ (t ranging from 1 to T), learning algorithm A, loss functional L, forecast horizon h, step Δt, training window size w_t (often fixed to a constant), and first test point t_0.

```
For $t$ ranging from $t_0$ to $T - h$ by steps $\Delta t$
    Training set: $\mathcal{D}_t = \{(x_s, y_{s+h})\}$, $s \in [t - w_t, t - h - 1]$
    Solution at $t$ is: $g_t = A(\mathcal{D}_t)$
    Test set: $\mathcal{T}_t = \{(x_s, y_{s+h})\}$, $s \in [t, t + \Delta t - 1]$
        Forecast at $s$: $g_t(x_s)$
        Loss at $s$: $l_s = L(g_t, (x_s, y_{s+h}))$
```

Output: the sequence of losses $\{l_t\}$, for $t \in [t_0, T - h]$

An illustration of the sliding training and test sets is shown in Figure 1.

The most important result of the sequential validation algorithm is the average loss, which can be compared across several algorithms. The individual losses are useful to esti-

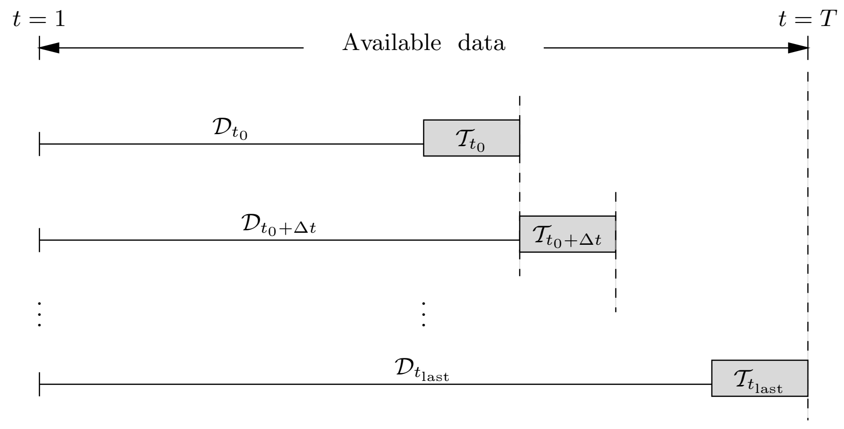
$t = 1$
$t = T$
Available data
$\mathcal{D}_{t_0}$
$\mathcal{T}_{t_0}$
$\mathcal{D}_{t_0+\Delta t}$
$\mathcal{T}_{t_0+\Delta t}$
$\mathcal{D}_{t_{\text{last}}}$
$\mathcal{T}_{t_{\text{last}}}$

Figure 1: *Sliding training and test sets that arise in the sequential validation procedure, where $\mathcal{D}_t$ denotes a training set (up to time t), $\mathcal{T}_t$ denotes a test set (starting from t), t_0 is the time of the first test point, and Δ_t is the increment between iterations. We assume here a training window size w_t of "infinite length", i.e. which uses all available past data.*

mate confidence intervals around the average loss or around differences in average loss (see section 3.3.2).

In the sequential validation algorithm we would in general prefer to choose $\Delta t = 1$ but larger values allow to save computations (in proportion to the value of Δt). The choice of the training window size w_t depends on the degree of non-stationarity expected (or estimated) from the particular data sequences. The most common choices are $w_t = \min(w_0, t)$ (a fixed value) and $w_t = t$ (use all the available data). Using sequences shorter than t may be justified when the conditional distribution of the data changes so much with t that old training pairs to hurt generalization to new cases. Note that unless w_t is constant the amount of training data may change as t increases, thus usually requiring an adaptive model selection algorithm.

3.1 Sequential Validation for Model Selection

Two levels of the sequential validation algorithm may be considered. At the topmost level, sequential validation may be used to estimate and compare the generalization performance of learning algorithms (including different model selection algorithms). At a lower level, it can be used within a learning algorithm in order to perform some form of model selection (more details in the experimental section). In particular, at time t, the sequence of past losses l_s for $s \in [t_0, t - h]$ for different algorithms (or solutions obtained with different complexities, or variable subsets) are available to perform a data-driven model selection. Sequential validation can also be used as a model selection procedure: the function class associated with the lowest validation error up to t is chosen for the forecast at t. However, in our experiments we generally found that cross-validation (within the training set) yielded better results for choosing the number of input variables.

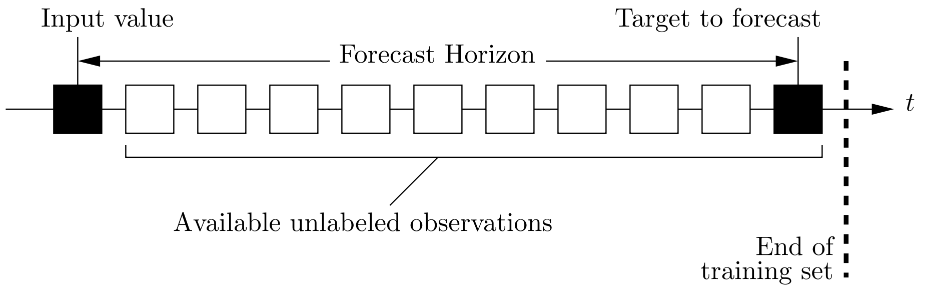

Figure 2: *A natural source of unlabeled data for time series forecasting with an horizon arises at the end of the training set.*

3.2 Natural Source of Unlabeled Data

Inspection of the sequential validation algorithm quickly reveals that at time t there are h input vectors x_s ($s \in [t-h+1, t]$) which cannot be associated with a corresponding target output y_{s+h}. The idea of the proposed extension is to use these **unlabeled points** to form the unlabeled set U required in the metric methods (to compute d_U, as in eq. (1)). This phenomenon is illustrated in Figure 2.

It is also interesting to note that this unlabeled set includes in particular the input for the next test point, x_t. This suggests that a method that uses x_t for model selection is actually doing a form of **transduction**. Vapnik introduced in 1982 (see also Vapnik, 1998) the principle of **transductive inference**, which differs from the usual **inductive inference** principle in that the learner chooses a solution based not only the training set but also on the input values of the test point(s). Here, this is particularly true when the sequential validation step Δt is chosen equal to 1 (which is however more computationally costly); otherwise, only one out of Δt of the test points would be in U.

Why would it be useful to use the metric model selection methods with the future test points as unlabeled data? The intuition is simply that these are the data points that we care about: this is where we most want to reject functions that "misbehave".

3.3 Time-Series Transduction Experiments

3.3.1 Experimental Setup

To verify the potential of metric model selection methods in time-series forecasting applications, we performed feature-selection experiments using artificially-generated data in a controlled setting. Our goal is to compare model selection algorithms (in this case the metric method ADJ against cross-validation) on the set of progressively more complex models that arise in forward (stepwise) feature selection.

Data Generation The artificial data series are generated from the class of linear autoregressive $AR(K)$ models, where given a fixed coefficients vector $\alpha \equiv (\alpha_0, \ldots, \alpha_K)'$ and initial conditions $y_{-1}, y_{-2}, \ldots, y_{-K}$, we have the process

$$y_t = \alpha_0 + \sum_{k=1}^{K} \alpha_k y_{t-k} + \epsilon_t, \quad t \geq 0. \tag{5}$$

with $\epsilon_t \sim N(0, \sigma^2)$ i.i.d. Gaussian noise.[1]

To simplify matters and ease analysis, we restrict the generating models to the specific form $y_t = \alpha + \alpha y_{t-K} + \epsilon_t$, where in our experiments $K = 1, 2, 3$.

Task Description We seek to forecast the series $\{y_t\}$ at horizon h, given the realizations of the past $\tilde{K}$ series values (we do not impose that $\tilde{K}$ be equal to the order K of the generating process). One typically considers a *point forecast*, or in other words, at a given time t and given the values of $\{y_t, y_{t-1}, \ldots, y_{t-\tilde{K}+1}\}$, one seeks an estimator of $E[y_{t+h}|\mathcal{I}_t]$. However, in our experiments, we shall consider an "integrated" forecast, consisting of the sum of the series values over the horizon, $y_{t+1} + y_{t+2} + \cdots + y_{t+h}$. We shall then seek an estimator of

$$E[y_{t+1} + y_{t+2} + \cdots + y_{t+h}|\mathcal{I}_t].$$

In many applications this type of forecast can be interpreted more naturally in terms of the underlying problem variables; for instance, given a financial series of (log) returns, the integrated forecast corresponds to the estimated total portfolio (log) return over the horizon. Obviously, at horizon $h = 1$, the integrated forecast is equivalent to the point forecast.

We shall consider the class of $AR(\tilde{K})$ models to make this forecast. This is equivalent to estimating the coefficients $\hat{\beta} \equiv (\hat{\beta}_0, \ldots, \hat{\beta}_{\tilde{K}})'$ corresponding to the model

$$\sum_{j=1}^{h} y_{t+j} = \beta_0 + \sum_{k=1}^{\tilde{K}} \beta_k y_{t-k+1} + \epsilon_t,$$

where ϵ_t is i.i.d. Gaussian noise.

The estimation of $\hat{\beta}$, for a fixed $\tilde{K}$, is easily performed analytically using the ridge estimator $\hat{\beta}^* = (X'X + \lambda I)^{-1} X'Y$, where X is the matrix of regressors, Y is the (column-)vector of targets, and I is the identity matrix.[2] This estimator implicitly uses a squared-error loss function, which is appropriate for our task of estimating a conditional expectation.

Feature Selection The role of feature selection here (see Algorithm 1) is to decide which α_k are significant and should be included in the regression. To this end, we use a standard forward stepwise selection algorithm, in which the individual features are the lagged series values, $y_{t-k}, k = 0, \ldots, \tilde{K} - 1$. Forward selection proceeds incrementally, starting from the mean (the lowest-complexity model that we are willing to consider), and at each step adds the feature that minimizes the training error. At a given time step t, we have the following sequence of models produced by the algorithm,

$$\{gt^{(0)}, g_t^{(1)}, \ldots, g_t^{(\tilde{K})}\},$$

1. We also performed experiments with $\epsilon_t \sim t(5)$, to evaluate the performance in the presence of an "uglier" fat-tailed distribution; see the results in Section 4.1.

2. In our procedure, we do not penalize the mean estimator $\hat{\beta}_0$; hence, the mean is estimated without bias.

where $g_t^{(k)}$ is the estimated regression model containing the k "best" features according to forward selection (which are not necessarily the first k lagged series values). The model $g_t^{(0)}$ is simply the mean on the current training set (obtained from $\mathcal{I}_t$).

We observe that this sequence of models forms a total order with respect to complexity, and is thence *amenable to selection by metric methods*. We exploit this crucial property, which arises naturally from the nature of the forward selection algorithm, in the experiments.

Experimental Plan The experiments measure the relative ability of 10-fold cross-validation versus metric model selection (in this case, ADJ) to select among the sequence of models produced by stepwise selection. We compare the methods across a whole spectrum of parameters, i.e. all permutations of:

- Forecasting horizon $h = \{1, 2, 5, 10, 15\}$
- Generating model AR order $K = \{1, 2, 3\}$
- Generating model coefficient magnitude $\alpha = \{0.1, 0.3, 0.5, 0.7, 0.9\}$. This coefficient controls the series *signal-to-noise ratio*; $\alpha = 0.1$ yields series very close to white noise, whereas series with $\alpha = 0.9$ exhibit much more structure.

Each triplet $\langle \text{horizon}, \text{AR order}, \text{magnitude} \rangle$ is henceforth called an *experiment*.

We fix the maximum model order $\tilde{K} = 10$, and a constant training window size $w_t = 75 = t_0$, making this a challenging task. The sequential validation increment is $\Delta_t = 10$, and the total length of each generated series is 1000 observations. In addition, each "basis" model $g_t^{(k)}$ is estimated with a small ridge penalty $\lambda = 10^{-\frac{1}{4}}$. (This hyperparameter was not tuned extensively, but empirically produced quite reasonable results.)

3.3.2 Statistical Methodology

We compare the performance of two models on a given experiment by a usual paired t-test on their mean-squared error difference. However, the results of individual experiments (e.g. across different horizons) cannot be pooled arbitrarily, since the expected error distribution is quite different across experiments. For instance, we *expect a priori* the MSE to be higher when forecasting across a longer horizon, given a stationary underlying generating process.

To perform a valid statistical test of the performance difference between methods *across experiments*, it is necessary to normalize the distribution of paired differences *within each experiment* to have unit standard deviation, before pooling the observations across experiments, and then performing the statistical test.

More specifically, suppose we perform M experiments, each one with N_m test points. Let $e_i^m, m = 1, \ldots, M, i = 1, \ldots, N_m$ be the squared error *differences* between two methods we wish to compare (e.g. cross-validation against ADJ in our case). The first step is to normalize the distribution of error differences to unit standard deviation,

$$\tilde{e}_i^m = \frac{e_i^m}{\sqrt{\hat{\sigma}^2(e^m)}}, \tag{6}$$

where the variance estimator $\hat{\sigma}^2(e^m)$ is described below. Then we compute the overall mean difference $\bar{e}$ and standard error $\hat{\sigma}_{\bar{e}}$ as

$$\bar{e} = \frac{\sum_{m=1}^{M} \sum_{i=1}^{N_m} \tilde{e}_i^m}{\sum_{m=1}^{M} N_m}, \qquad \hat{\sigma}_{\bar{e}} = \frac{1}{\sqrt{\sum_{m=1}^{M} N_m}}. \tag{7}$$

Throughout this section, the so-obtained mean difference $\bar{e}$ is termed *normalized MSE difference.*

Estimation of $\sigma(e^m)$ The question left open is the estimation of the standard deviation of the error-difference distribution within a single experiment. The usual estimator cannot be used here for it rests upon an i.i.d. assumption, whereas the series we consider exhibit mild to strong autocorrelation patterns. This autocorrelation is induced, on the one hand, by the problem structure, and on the other hand by the sequential validation testing procedure.[3]

To properly estimate the variance, we use the Newey–West estimator well-known to econometricians (Newey and West, 1987, Diebold and Mariano, 1995, Campbell et al., 1997), which in addition to being consistent, has the desirable property of being robust at small sample sizes,[4]

$$\hat{\sigma}^2(e^m) = \hat{\gamma}_0^m + 2\sum_{j=1}^{q} \frac{q-j}{q} \hat{\gamma}_j^m, \tag{8}$$

where q is the maximum lag length to be considered,[5] and $\hat{\gamma}_j^m$ is the empirical lag-j autocovariance,

$$\hat{\gamma}_j^m \stackrel{\text{def}}{=} \frac{1}{N_m - j} \sum_{i=1}^{N_m - j} (e_i^m - \bar{e}^m)(e_{i+j}^m - \bar{e}^m), \tag{9}$$

with $\bar{e}^m$ the sample mean.

3.3.3 Experimental Results

Figure 3 presents a summary of the experiments described above, comparing cross-validation against ADJ. The left plot outlines the effect of the series signal-to-noise (SNR) ratio (for which the generating model AR coefficient magnitude are a proxy) on performance. At very low SNR, the series being essentially white noise, both methods perform about equally poorly (worse, in fact, than a naive constant model (not shown on the figure)). At the other end of the spectrum, at high coefficient values, cross-validation performs, overall, significantly better than ADJ. However, the opposite picture emerges at small but significant coefficient values, where ADJ significantly beats cross-validation. We conjecture that at these moderate SNR levels, the intrinsic variance of the choice made by cross-validation causes costly mistakes, whereas a less-variable (albeit biased) method such as ADJ can pick out important structures without being swamped by the noise level.

3. Since successive training sets in sequential validation tend to highly overlap, the trained models are generally very correlated—especially at small step sizes Δ_t—thence inducing correlation in the error structure.
4. It guarantees the positive-definiteness of the estimated covariance matrix in the multivariate case.
5. This must scale with the sample size for the estimator to be consistent, but not too rapidly.

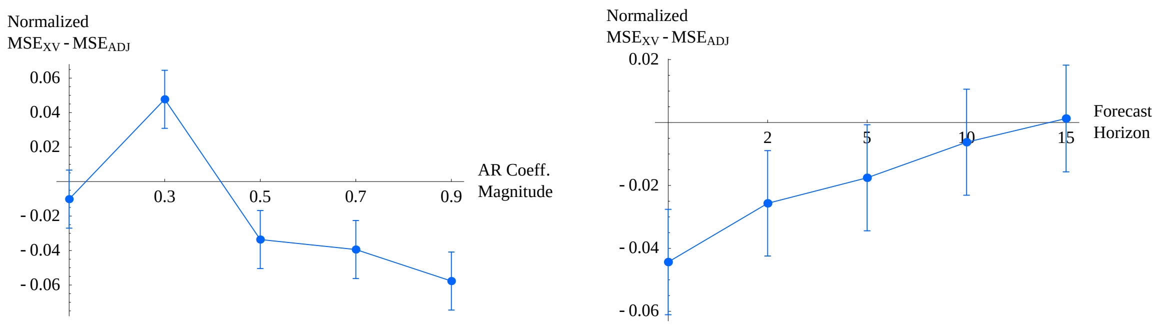

Figure 3: ***Left:*** *Normalized MSE difference between the models chosen by cross-validation and ADJ, as a function of the magnitude of the AR coefficients (across all forecast horizons and generating model order). The error bars represent 95% confidence intervals on the mean difference (normalized as explained in the text).* ***Right:*** *Same measure, as a function of the forecast horizon; we note that even with* ***extremely few*** *unlabeled observations (one or two), ADJ does not lose catastrophically against cross-validation, which is very surprising; the two methods become essentially equivalent for longer horizons.*

The right plot in Figure 3 is, in some ways, more surprising: first, the expected outcome shows a steady improvement in the performance of ADJ with respect to cross-validation as the forecast horizon increases, as a result of the increase in the number of unlabeled observations that ADJ can use to make its choice. But the unexpected outcome is, relatively speaking, **how well ADJ performs given extremely few unlabeled observations** (one or two); recall that these observations are used to form a Monte Carlo approximator of an expectation (*c.f.* eq. 1), and that so few observations are sufficient to make a reasonable model selection choice in this context strikes us as a surprise.

Moreover, we can count the number of experiments for which each method statistically significantly beats the other; a kind of model selection tournament (we shall take $p \leq 0.05$ as the significance level). The results comparing ADJ to cross validation are shown in Table 1. The hypothesis about the behavior of each method at a given series SNR finds more confirmation; a further surprise emerges from the horizon data, where we find that ADJ sometimes **significantly beats** cross-validation even at very small forecast horizon (i.e. using extremely few unlabeled points). The two methods become indistinguishable at longer horizons.

4. “Meta” Model Selection

The motivation for this other extension to the metric model selection methods follows from several years of working with various model selection methods and frustratingly comparing them against cross-validation. Cross-validation does not always work but it almost always performs quite well. However, it tends to have higher variance (in the sense of larger variations in error) than complexity penalization methods. We also know that it is almost unbiased (it is unbiased for training with a bit less examples than what is actually

Table 1: *"Tournament" results comparing cross-validation against ADJ for individual experiments. A "win" indicates that the corresponding method beats the other at a statistical significance level of at least 95% ($p \leq 0.05$) on the MSE criterion. A blank stands for zero. The results corroborate those of Figure 3.*

	XV Wins	ADJ Wins	Total Experiments
Overall	17	7	75
AR Coeff = 0.01			15
AR Coeff = 0.03		7	15
AR Coeff = 0.05	1		15
AR Coeff = 0.07	6		15
AR Coeff = 0.09	10		15
Horizon = 1	7	1	15
Horizon = 2	6	1	15
Horizon = 5	3	3	15
Horizon = 10	1	2	15
Horizon = 15			15

available).[6] Since it is usually almost as good (and often better) than these complexity penalization methods (including the metric methods), it must mean that these other methods must have smaller variance (and none of them is guaranteed to be unbiased, so they are likely to be biased). Can we take advantage of this situation, whereby one method is more biased but has less variance than the other.

In this paper we have just begun to explore this opportunity. Let us call g^{xv} the solution obtained by cross-validation and g^p the solution obtained by some form of complexity penalization, for a particular training set. A simple-minded combination algorithm is the following: if, for a given test point x, the absolute difference $|g^{xv}(x) - g^p(x)|$ is "large", then trust g^p, else trust g^{xv}. The intuition for this heuristic rule is that a large difference in function value more likely indicates that the cross-validatory choice is wrong, owing to its large variance. This leaves open the question of choosing the proper threshold. A more sophisticated (and better grounded) algorithm for the squared loss, which we have tested in the experiments is shown in Algorithm 3.

This algorithm is based on the idea of the logistic regression: it assigns a weight $w_s(\beta)$ to the cross-validation model (and $1 - w_s(\beta)$ to the ADJ model) based on a quadratic function of the difference $g^{xv}(x) - g^p(x)$;[7] the use of the sigmoid ensures that the weights are always between 0 and 1. Contrarily to traditional logistic regression, the coefficient vector β for the weights is obtained by directly optimizing a squared-loss criterion, estimated from the *past test observations* (i.e. those for $s \leq t$, available without cheating from the sequential validation procedure).

6. We are talking about the bias of an estimator of generalization error. However, for most model selection methods, the only bias we care about is not in the value of the estimator but only of how it ranks different hypotheses.
7. The choice of this functional form is somewhat arbitrary, but yields empirically good results.

```
Algorithm 3 Logistic Meta Model Selection
Input at $t$: the sequence of solutions $g_s^{xv}$ and $g_s^p$, respectively for the cross-validation and complexity penalization selected models, and the data sequence $\{(x_s, y_s)\}$ for $s \leq t$.
  1. Let $d_s := g_s^{xv}(x_s) - g_s^p(x_s)$
  2. Let $w_s(\beta) := 1/(1 + \exp(-(\beta_0 + \beta_1 d_s + \beta_2 d_s^2)))$
  3. Let $C(\beta) := \sum_{s \leq t-h}(w_s(\beta) g_s^{xv} + (1 - w_s(\beta)) g_s^p - y_{s+h})^2$
  4. Let $\beta^* := \text{argmin}_\beta C(\beta)$
Output at $t$: the solution $g_t := w_t(\beta) g_t^{xv} + (1 - w_t(\beta)) g_t^p$.
```

Table 2: *"Tournament" results comparing the logistic combination against, respectively cross-validation alone and ADJ alone, for individual experiments. A "win" indicates that the corresponding method beats the other at a statistical significance level of at least 95% ($p \leq 0.05$) on the MSE criterion. A blank stands for zero.*

	Logis Wins	XV Wins	Logis Wins	ADJ Wins	Total Exp.
Overall	6	1	15	1	75
AR Coeff = 0.01			1		15
AR Coeff = 0.03	6				15
AR Coeff = 0.05		1	3	1	15
AR Coeff = 0.07			3		15
AR Coeff = 0.09			8		15
Horizon = 1	1		7		15
Horizon = 2	1		6		15
Horizon = 5	3	1	2		15
Horizon = 10	1				15
Horizon = 15				1	15

Like Boosting (Freund and Schapire, 1996) and other model combination algorithms, this algorithm combines multiple predictors, and it does so in a way that gives more weight to the best predictor. However, the weight changes from example to example, as a learned function of the difference between the two predictors.

4.1 Meta Model Selection Experiments

The same experimental setup as described in section 3.3.1 was used to compare the logistic combination rule against cross-validation alone and ADJ alone. "Tournament" results are shown in Table 2. The most significant observation is that the logistic combination **almost always performs better** than either method taken alone. As the table shows, across all experiments, the logistic combination loses only once to cross-validation and ADJ, whereas it wins more frequently against them.

Table 3: *Number of times that each algorithm* **beats**/*is beaten by cross-validation, under both a normal and a t distribution, in a total of 75 experiments (described in the text). "Beating", as in the previous tournament results, is to exhibit a statistically significantly better performance, at the* $p = 0.05$ *level.*

	ADJ		SEB	
	$N(0,1)$	$t(5)$	$N(0,1)$	$t(5)$
Logistic combination	**6**/*1*	**3**/*3*	**12**/*6*	**15**/*6*
Simple average	**15**/*7*	**11**/*7*	**28**/*4*	**35**/*3*

Table 3 shows "tournament" results against cross-validation in the same experimental setting as described above (each tournament made up of the 75 experiments), in various conditions:

- **Varying the combination algorithm**. We tried both the logistic combination (Algorithm 3) between the cross-validatory choice and an alternative metric-method choice, and an equal-weight combination (simple average) between the two; i.e. in the notation of Algorithm 3, we have the solution $g_t := \frac{1}{2} g_t^{xv} + \frac{1}{2} g_t^p$.
- **Varying the metric model selection method**. We tried both ADJ and the *Smallest Eigenvalue Bound* (SEB) method of Chapelle et al. (2002).
- Process innovations ϵ_t either following a $N(0,1)$ distribution or a fat-tailed Student $t(5)$ distribution.

These results suggest a few interesting conclusions: 1) The simple average combination performs surprisingly well (even better, in most cases, than the logistic combination), and is quite easier to implement. 2) With ADJ, the logistic combination appears more robust (loses less often to cross-validation alone). 3) Combining SEB and cross-validation performs very well, better than combining ADJ and cross-validation; we conjecture that SEB has an even smaller variance than ADJ (albeit with a possibly worse bias), and this should be the basis of future investigations.

Finally, Figure 4 illustrates typical cases of the weight attributed to the cross-validation model resulting from the logistic combination (as a function of $g^{xv} - g^p$). (The alternative ADJ-selected model gets the opposite weight). It confirms the validity of the intuition outlined above: in case of "small" differences (but with a bias empirically estimated from the data) between g^{xv} and g^p, choose cross-validation, otherwise choose ADJ. It should be noted in passing that the steepness of the transition in the weights depends in large part on the "aggressiveness" of the numerical optimization performed to the parameters; when pushing the optimization to its limit, one usually ends up with very large weights that yield an abrupt transition, an effect that is not quite desirable from the viewpoint of numerical stability.

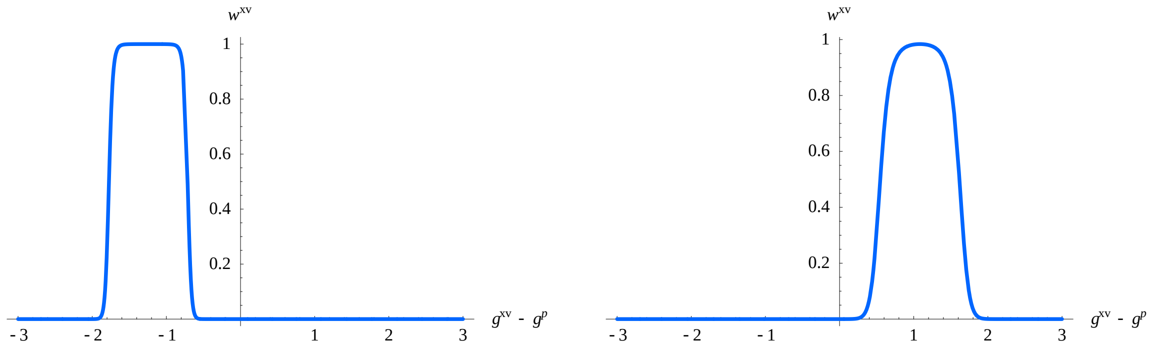

Figure 4: *Examples of the weight $w_s(\beta)$ given to the cross-validation model (c.f. Algorithm 3), obtained by the logistic regression, as a function of the difference $g^{xv} - g^p$ evaluated at the test point.*

5. Proposed Extension when No Unlabeled Data is Available

The final extension to TRI, ADJ and ADA applies to the case where *no unlabeled data is available*, which occurs in many applications. The basic idea is to approximate the expected value of $d(f, g)$ using a *model of the input density* rather than with an average over unlabeled data,

$$\tilde{d}(f,g) \overset{\text{def}}{=} \psi \left(\int \hat{P}_X(x) L(f(x), g(x)) dx \right),$$

where $\hat{P}_X(x)$ is the model of the input density. In practice this integral may be difficult to compute analytically so we used a simple Monte-Carlo procedure based on sampling from the density model $\hat{P}_X$. This model may be derived from the training set input points $(x_1, \ldots, x_m)$ and/or prior knowledge about the input density. In our experiments we have used a Parzen windows density estimator (that only relies on the data),

$$\hat{P}_X(x) \overset{\text{def}}{=} \frac{1}{m} \sum_i \frac{e^{-0.5||x_i - x||^2/\sigma^2}}{(2\pi)^{n/2}\sigma^n},$$

where $x \in \mathcal{R}^n$. Such a model leaves the smoothing parameter to be chosen. Many methods have been proposed for setting the smoothing parameter of such non-parametric density estimators. In our experiments we have experimented with the effect of varying σ and we have used an asymptotically motivated estimator Scott (1992):

$$\sigma = \frac{1.144\sqrt{\widehat{\text{Var}}[X]}}{m^{1/5}}. \qquad (10)$$

where $\widehat{\text{Var}}[X]$ denotes sample variance of the inputs on the training set.

5.1 Extension of Theoretical Results

In Schuurmans and Southey (2002), several attractive theoretical results have been proved concerning the generalization performance of TRI, ADJ and ADA. Almost identical results can be proved for the extended algorithms that use an estimated input density $\hat{P}_X(x)$, with the following change: instead of characterizing the true generalization error (i.e. averaging the error over the true input density $P_X(x)$), those results characterize the generalization error measured with respect to the estimated input density $\hat{P}_X(x)$. The main theoretical results are briefly recalled here. Let us call xTRI, xADJ, and xADA the extended methods using $\hat{P}_X(x)$.

Proposition 1 Let $g_m = \text{argmin}_{g_i} \tilde{d}(g_i, P_{Y|X})$, and let g_l be the hypothesis selected by xTRI. If $m \leq l$ and $d_T(g_m, P_{Y|X}) \leq \tilde{d}(g_m, P_{Y|X})$ then $\tilde{d}(g_l, P_{Y|X}) \leq 3\tilde{d}(g_m, P_{Y|X})$.

The above tells us when xTRI cannot overfit too much.

Proposition 2 Let $g_m = \text{argmin}_{g_i} \tilde{d}(g_i, P_{Y|X})$, and let g_l be the hypothesis selected by xADJ. If $m \leq l$ and $\hat{d}(g_m, P_{Y|X}) \leq \tilde{d}(g_m, P_{Y|X}))$ then $\tilde{d}(g_l, P_{Y|X}) \leq (2+\frac{d_T(g_m, P_{Y|X})}{d_T(g_l, P_{Y|X})})\tilde{d}(g_m, P_{Y|X})$.

The above tells us when xADJ cannot overfit too much.

Proposition 3 Consider a hypothesis g_m, and assume that $d_T(g_l, P_{Y|X}) \leq \tilde{d}(g_l, P_{Y|X})$ for $l \leq m$, and $d_T(g_l, P_{Y|X}) \leq \hat{d}(g_l, P_{Y|X})$ for $l \leq m$. Then if $\tilde{d}(g_m, P_{Y|X}) \leq \frac{1}{3}\frac{d_T(g_l, P_{Y|X})^2}{\tilde{d}(g_l, P_{Y|X})}$ for $l < m$ (i.e. $\tilde{d}(g_m, P_{Y|X})$ is sufficiently small) it follows that $\hat{d}(g_m, P_{Y|X}) < \hat{d}(g_l, P_{Y|X})$ for $l < m$ and therefore xADJ will not choose any predecessor in lieu of g_m.

The above tells us when xADJ cannot underfit too much. See Schuurmans and Southey (2002) for proofs of equivalent results for the original methods, and detailed discussions. The same authors observe that the conditions required for these propositions to hold are not always satisfied in practice.

5.2 Experimental Results

5.2.1 Artificial Data Results

To ascertain the validity of the proposed extensions, we first performed experiments on artificial data, keeping the same overall framework as Schuurmans and Southey (2002). We tested the methods on data generated from the four following functions:

$$f_1(x) = \mathbf{1}_{x \geq 0.5} \qquad f_3(x) = \sin^2(2\pi x)$$
$$f_2(x) = \sin(1/x) \qquad f_4(x) = 32x - 275x^2 + 777x^3 - 892x^4 + 358x^5$$

Our class of nested models is the set of polynomials of fixed order,

$$g(x;\beta) = \sum_{j=0}^{m} \beta_j x^j,$$

where the order m depends on the size of the training set (described below). The models are trained to minimize a squared error criterion with weight decay; the members of the class vary according to the value of the weight-decay coefficient λ (a higher λ implies a reduced capacity). We restricted λ to the set $\{10^i : i = -5, -4.5, -4, \ldots, 2\}$, which was found to be adequate. The optimal parameter vector $\hat{\beta}^*$ is given by the well-known *ridge estimator*, $\hat{\beta}^* = (X'X + \lambda I)^{-1}X'Y$, where X is the matrix of regressors, Y is the (column-) vector of targets, and I is the identity matrix.

The artificial data sets are generated from the functions $f_1, \ldots, f_4$. The input density $P_X(x)$ is chosen as either uniform on the $[0, 1]$ interval ($U(0, 1)$), or normal with mean

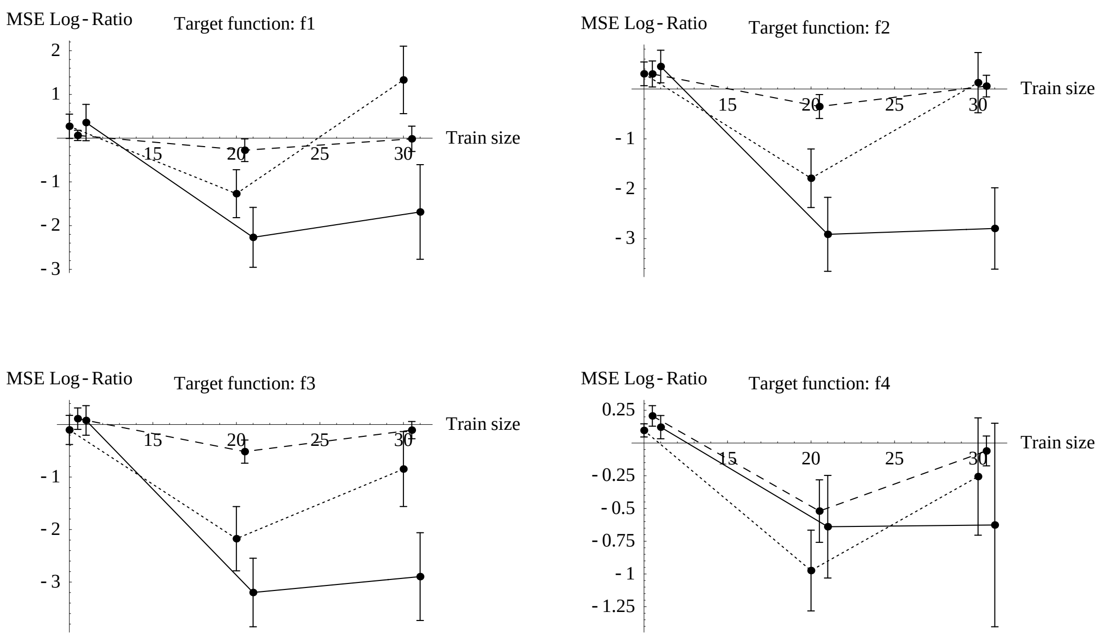

Figure 5: *Comparison between Parzen-generated unlabeled data and "true" unlabeled data, for xTRI/TRI (dotted line), xADJ/ADJ (dashed line), and xADA/ADA (solid line). Each point is the median test MSE log-ratio of 50 repetitions, with random training, unlabeled and test sets; the error bars represent 1 standard error on the median, computed by bootstrap resampling. The input distribution is $N(\frac{1}{2}, \frac{1}{12})$. Surprisingly, the Parzen-generated data sometimes performs* ***significantly better*** *(negative log-ratio) than unlabeled data from "true" $P_X(x)$, and never significantly worse.*

$\frac{1}{2}$ and variance $\frac{1}{12}$ ($N(\frac{1}{2}, \frac{1}{12})$, chosen to have the same mean and variance as the $U(0, 1)$ distribution). Additive noise with a distribution $N(0, 0.0025)$ is added in every case.

As in Schuurmans and Southey (2002), training sets of sizes $\{10, 20, 30\}$ are considered. The size of the unlabeled sets generated from $P_X(x)$ (for TRI, ADJ, and ADA) is fixed to 200. The same number of unlabeled examples are sampled from the Parzen estimator for xTRI, xADJ, and xADA. Finally, a single test set of size 5000 is generated to evaluate performance.

Figure 5 compares each method against 10-fold cross-validation. We observe that the proposed methods compare favorably to the originals, sometimes beating them statistically significantly.

Figure 6 shows the effect of the Parzen estimator bandwidth σ. Panel (a) shows the sensitivity of the median MSE of the three algorithms to σ; xADA is the least sensitive to the choice of σ. Panel (b) shows the sensitivity of the log-norm of the parameters (i.e. reflecting the complexity of the solution) with respect to σ. In most cases, as we expected, a higher σ yields simpler solutions. However, unexpectedly, we find in one case that larger σ yields larger parameters (for xADA).

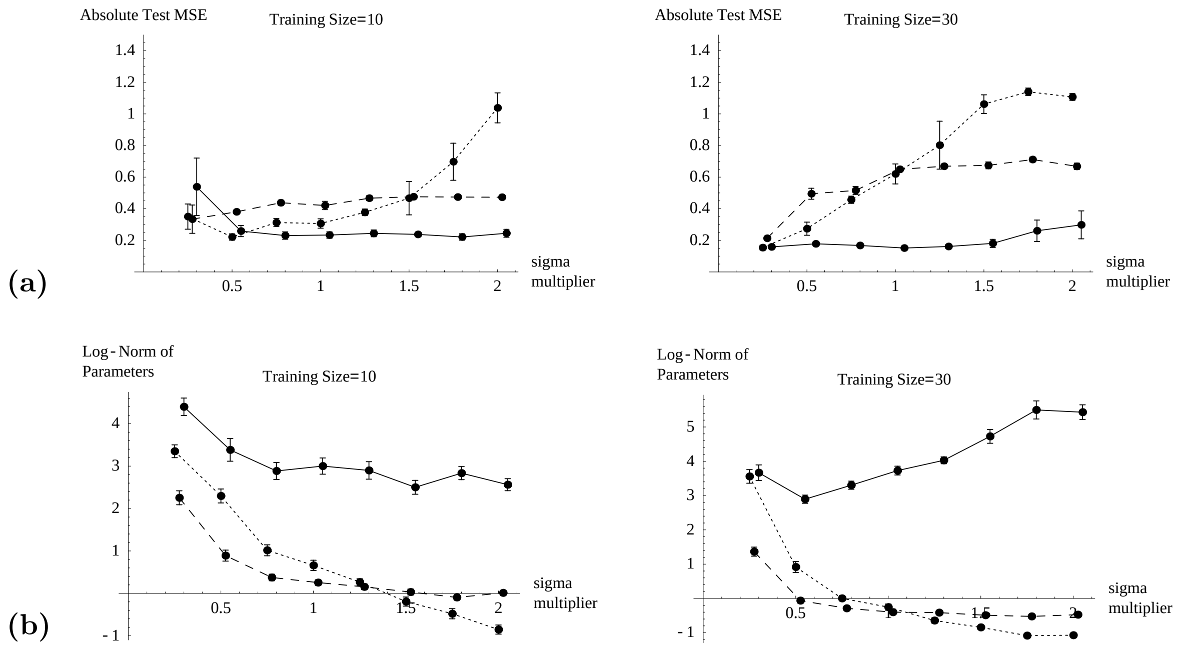

Figure 6: *Effect of Parzen estimator window width for xTRI (dotted line), xADJ (dashed line), and xADA (solid line).* ***Panel (a)*** *shows the median MSE performance; we observe that xTRI is very sensitive to the window width, xADJ less to, and that xADA is nearly insensitive to it.* ***Panel (b)*** *shows the log-norm of the parameters of the trained models; we note that the window width has a clear regularization effect for xTRI and xADJ.* ***In both cases****, the window width is given as a multiplicative factor applied to eq.* (10). *All results are computed for function f1 under a* $U(0,1)$ *input distribution, 50 repetitions. The error bars represent 1 standard error on the median and the mean respectively.*

5.2.2 Results on the Boston Housing Dataset

We also performed a series of experiments involving a combination of *feature selection* along with model selection on a non-artificial regression task (Boston Housing) from the UCI Machine Learning Repository. We used for this purpose an RBF network regularized with weight decay, with one RBF per training example:

$$g(z;\sigma) = \sum_{i=1}^{m} w_i \exp\left(-\frac{\|z-x_i\|^2}{2\sigma^2}\right), \tag{11}$$

where $\{x_i\}_{i=1}^m$ are the training set inputs, and σ is the RBF window width.[8] For a given weight decay λ, an analytical solution for the w_i that minimizes the squared error criterion is easily obtained (similar to the ridge estimator presented in section 5.2.1).

We used a standard forward stepwise feature selection procedure (as described in algorithm 1 and section 3.3.1) to obtain a *nested family* of models of increasing complexity. We

8. Not to be confused with the window width of the Parzen distribution used to sample unlabeled examples!

Table 4: *Comparing cross-validation (XVT), ADJ, and xADJ on the UCI Boston Housing dataset. The table reports the average Test MSE and standard errors, within the Forward Feature Selection setting described in the text.* Max. Var. *is the maximum number of features allowed in the forward selection procedure. To obtain standard errors, random (nonoverlapping) training (size=101), unlabeled (size=303), and test (size=102) sets are drawn,* Num. Rep. *times (which differs depending on* Max. Var. *for computation time reasons). The three rightmost columns give the p-values that the paired MSE difference between the indicated algorithms is zero, under the appropriate t-distribution. We note that plain ADJ is always significantly worse than either XVT and xADJ, and no significant difference is found between the latter two. The underlying Boston dataset contains 506 observations with no missing values. The learning algorithm is an RBF network with constant weight decay=10^{-2}, gaussian window width=1, and one RBF per training point.*

Max. Var.	Num. Reps	MSE XVT (std err)	MSE ADJ (std err)	MSE xADJ (std err)	$p_{\text{XVT}-\text{ADJ}}$	$p_{\text{XVT}-\text{xADJ}}$	$p_{\text{ADJ}-\text{xADJ}}$
5	100	0.000204 (0.000013)	0.000229 (0.000013)	0.000210 (0.000013)	**0.001** ⋆	0.264	**0.007** ⋆
10	50	0.000190 (0.000019)	0.000238 (0.000022)	0.000202 (0.000019)	**0.003** ⋆	0.211	**0.004** ⋆
18	40	0.000158 (0.000017)	0.000191 (0.000021)	0.000173 (0.000019)	**0.023** ⋆	0.131	**0.028** ⋆

then use a model selection algorithm to select the best-performing model among this family. Table 4 compares the performance of cross-validation (XVT), ADJ, and xADJ on the Boston dataset (506 observations with no missing values, 18 features, 1 continuous target). The difficulty of the model selection problem is controlled by the maximum allowed number of features; we compared the performance allowing a maximum of 5, 10, and 18 features.

The results indicate that both cross-validation and xADJ perform, on average, significantly better than plain ADJ. This is surprising, since the latter has access to more information (303 unlabeled observations from the "true" distribution). Furthermore, xADJ never performs significantly worse than cross-validation. In all cases, 300 unlabeled observations were generated from the Parzen distribution for xADJ. The differences between ADJ and xADJ could just be due a statistical fluctuation occuring with the particular data used in the experiments. However, since we do not fully understand the nature of the generalization improvement brought by ADJ there could also be another explanation, which future work should investigate.

6. Conclusions

We have presented three new model and feature selection methods, with the goals of improving and extending the applicability of metric-based methods.

A first surprising result is that ADJ can work with time-series data (and not be beaten by cross-validation in the majority of cases) in a transductive fashion while using only as few as 1 to 10 unlabeled examples (whereas one would have expected that with so few examples the complexity correction computed by ADJ would be very unreliable).

A second surprising result is that using non-parametric density estimators, ADJ can be succesfully applied even when an independent source of unlabeled data is not available.

Finally, a third and exciting result is the benefit gained by combining the models chosen by cross-validation and another model selection method with lower variance, such as ADJ and SEB. We conjecture that the significant improvements that we have observed are attributable to the different error profiles of the respective methods. This should be explored more deeply in future work. A theoretical analysis of the first two results also poses challenges for future investigations.

References

Y. Bengio and D. Schuurmans. Special issue on new methods for model selection and model combination, 2002. *Machine Learning*, 48(1).

J.Y. Campbell, A. W. Lo, and A.C. MacKinlay. *The Econometrics of Financial Markets*. Princeton University Press, Princeton, 1997.

O. Chapelle, V. Vapnik, and Y. Bengio. Model selection for small-sample regression. *Machine Learning Journal*, 48(1):9–23, 2002.

Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 416–422, 2001.

F. X. Diebold and R. S. Mariano. Comparing predictive accuracy. *Journal of Business and Economic Statistics*, 13(3):253–263, 1995.

D. Foster and E. George. The risk inflation criterion for multiple regression. *Annals of Statistics*, 22:1947–1975, 1994.

Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of Thirteenth International Conference*, pages 148–156, 1996.

W. Newey and K. West. A simple, positive semi-definite, heteroscedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55:703–708, 1987.

T. Poggio and F. Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, 247:978–982, 1990.

J. Rissanen. Stochastic complexity and modeling. *Annals of Statistics*, 14:1080–1100, 1986.

D. Schuurmans. A new metric-based approach to model selection. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-97)*, pages 552–558, 1997.

D. Schuurmans and F. Southey. Metric-based methods for adaptive model selection and regularization. *Machine Learning*, 48(1):51–84, 2002.

D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization.* Wiley, New York, 1992.

V. Vapnik. *Statistical Learning Theory.* Wiley, Lecture Notes in Economics and Mathematical Systems, volume 454, 1998.

V.N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, Berlin, 1982.