Neural Computation Manuscript # 2045

Gradient-Based Optimization of Hyper-Parameters

Yoshua Bengio
Département d'informatique et recherche opérationnelle
Université de Montréal
C.P. 6128 Succ. Centre-Ville, Montréal, Québec, Canada, H3C 3J7
bengioy@iro.umontreal.ca

September 30, 1999

Abstract

Many machine learning algorithms can be formulated as the minimization of a training criterion which involves a hyper-parameter. This hyper-parameter is usually chosen by trial and error with a model selection criterion. In this paper we present a methodology to optimize several hyper-parameters, based on the computation of the gradient of a model selection criterion with respect to the hyper-parameters. In the case of a quadratic training criterion, the gradient of the selection criterion with respect to the hyper-parameters is efficiently computed by back-propagating through a Cholesky decomposition. In the more general case, we show that the implicit function theorem can be used to derive a formula for the hyper-parameter gradient involving second derivatives of the training criterion.

1 Introduction

Machine learning algorithms pick a function from a set of functions $\mathcal{F}$ in order to minimize something that cannot be measured, only estimated, that is the expected generalization performance of the chosen function. Many machine learning algorithms can be formulated as the minimization of a **training criterion** which involves a hyper-parameter, kept fixed during this minimization. For example, in the regularization framework (Tikhonov and Arsenin, 1977; Poggio, Torre and Koch, 1985), one hyper-parameter controls the strength of the penalty term: a larger penalty term reduces the "complexity" of the resulting function (it forces the solution to lie in a "smaller" subset of $\mathcal{F}$). A common example is *weight decay* (Hinton, 1987), used with neural networks and linear regression (also known as ridge regression (Hoerl and Kennard, 1970), in that case), to penalize the L2-norm of the parameters. Increasing the penalty term (increasing the weight decay hyper-parameter) corresponds to reducing the *effective capacity* (Guyon et al., 1992) by forcing the solution to be in a zero-centered hyper-sphere of smaller radius, which may improve generalization. A regularization term can also be interpreted as an a-priori probability distribution on $\mathcal{F}$: in that case the weight decay is a scale parameter (e.g., inverse variance) of that distribution.

A **model selection criterion** can be used to select hyper-parameters, more generally to compare and choose among models which may have a different capacity. Many model selection criteria have been proposed in the past (Vapnik, 1982; Akaike, 1974; Craven and Wahba, 1979). When there is only a single hyper-parameter one can easily explore how its value affects the model selection criterion: typically one tries a finite number of values of the hyper-parameter and picks the one which gives the lowest value of the model selection criterion.

In this paper we present a methodology to simultaneously *select many hyper-parameters using the gradient of the model selection criterion with respect to the hyper-parameters*. This methodology can be applied when some differentiability and continuity conditions of the training criterion are satisfied. The use of multiple hyper-parameters has already been proposed in the Bayesian literature: one hyper-

parameter per input feature was used to control the prior on the parameters associated to that input feature (MacKay and Neal, 1994; Neal, 1998). In this case, the hyper-parameters can be interpreted as scale parameters for the prior distribution on the parameters, for different directions in parameter space. In Sections 2, 3 and 4, we explain how the gradient with respect to the hyper-parameters can be computed. In the conclusion, we briefly describe the results of preliminary experiments performed with the proposed methodology (described in more details in (Latendresse and Bengio, 1999; Bengio and Dugas, 1999)), and we raise some important open questions concerning the kind of "over-fitting" that can occur with the proposed methodology.

2 Objective Functions for Hyper-Parameters

We are given a set of independent data points, $D = \{z_1, \ldots, z_T\}$, all generated by the same unknown distribution $P(Z)$. We are given a set of functions $\mathcal{F}$ indexed by a parameter $\theta \in \Omega$ (i.e., each value of the parameter θ corresponds to a function in $\mathcal{F}$). In our applications we will have $\Omega \subseteq \mathcal{R}^s$. We would like to choose a value of θ that minimizes the expectation $E_Z(Q(\theta, Z))$ of a given loss functional $Q(\theta, Z)$. In supervised learning problems, we have input/output pairs $Z = (X, Y)$, with $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, and θ is associated to a function f_θ from $\mathcal{X}$ to $\mathcal{Y}$. For example, we will consider the case of the quadratic loss, with real-valued vectors $\mathcal{Y} \subseteq \mathcal{R}^m$ and $Q(\theta, (X, Y)) = \frac{1}{2}(f_\theta(X) - Y)'(f_\theta(X) - Y)$. Note that we use the letter θ to represent parameters and the letter λ to represent hyper-parameters.

In the next section, we will provide a formulation for the cases in which Q is quadratic in θ (e.g., quadratic loss with constant or affine function sets). In section 4, we will consider more general classes of functions and loss, which may be applied to the case of multi-layer neural networks, for example.

2.1 Training Criteria

In its most general form, a **training criterion** C is any real-valued function of the set of empirical losses $Q(\theta, z_i)$ and of some hyper-parameters λ:

$$C(\theta, \lambda, D) = c(\lambda, Q(\theta, z_1), Q(\theta, z_2), \ldots, Q(\theta, z_T))$$

Here λ is assumed to be a real-valued vector $\lambda = (\lambda_1, \ldots, \lambda_q)$. The proposed method relies on the assumption that C is continuous and differentiable almost everywhere with respect to θ and λ. When the hyper-parameters are fixed, the learning algorithm attempts to perform the following minimization:

$$\theta(\lambda, D) = \text{argmin}_\theta C(\theta, \lambda, D) \tag{1}$$

An example of training criterion with hyper-parameters is the following:

$$C = \sum_{(x_i, y_i) \in D} w_i(\lambda)(f_\theta(x_i) - y_i)^2 + \theta' A(\lambda)\theta \tag{2}$$

where the hyper-parameters provide different quadratic penalties to different parameters (with the matrix $A(\lambda)$), and different weights to different training patterns (with $w_i(\lambda)$), (as in (Bengio and Dugas, 1999; Latendresse and Bengio, 1999)).

2.2 Model Selection Criteria

The **model selection criterion** E is a criterion that is used to select hyper-parameters or more generally to choose one model among several models. Ideally, it should be the expected generalization error (for a fixed λ), but $P(Z)$ is unknown, so many alternatives have been proposed, which are either approximations, bounds, or empirical estimates. Most model selection criteria have been proposed for selecting a single hyper-parameter that controls the "complexity" of the class of functions in which the learning algorithms finds a solution, e.g. the minimum description length principle (Rissanen, 1990), structural risk minimization (Vapnik, 1982), the Akaike Information Criterion (Akaike, 1974), or the

generalized cross-validation criterion (Craven and Wahba, 1979). Another type of criteria are those based on held-out data, such as the cross-validation estimates of generalization error. These are almost unbiased estimates of generalization error (Vapnik, 1982) obtained by testing f_θ on data not used to choose θ. For example, the K-fold cross-validation estimate uses K partitions of D, $S_1^1 \cup S_2^1$, $S_1^2 \cup S_2^2$, ... and $S_1^K \cup S_2^K$:

$$E_{cv}(\lambda, D) = \frac{1}{K} \sum_i \frac{1}{|S_2^i|} \sum_{z_t \in S_2^i} Q(\theta(\lambda, S_1^i), z_t).$$

When θ is fixed, the empirical risk $\frac{1}{T} \sum_t Q(\theta, z_t)$ is an unbiased estimate of the generalization error of f_θ (but it becomes an optimistic estimate when θ is chosen to minimize the empirical risk). Similarly, when λ is fixed, the cross-validation criterion is an almost unbiased estimate (when K approaches $|D|$) of the generalization error of $\theta(\lambda, D)$. When λ is chosen to minimize the cross-validation criterion, this minimum value also becomes an optimistic estimate. Likewise, when there is a greater diversity of values $Q(f_{\theta(\lambda,D)}, z)$ that can be obtained for different values of λ, there is more risk of **over-fitting the hyper-parameters**. In this sense, the use of hyper-parameters proposed in this paper can be very different from the common use in which a hyper-parameter helps to control over-fitting. Instead, a blind use of the extra freedom brought by many hyper-parameters could deteriorate generalization.

3 Optimizing Hyper-Parameters for a Quadratic Training Criterion

In this section we analyze the simpler case in which the training criterion C is a quadratic polynomial of the parameters θ. The dependence on the hyper-parameters λ can be of higher order, as long as it is continuous and differentiable almost everywhere (see for example (Bottou, 1998) for more detailed technical

conditions sufficient for stochastic gradient descent):

$$C = a(\lambda) + b(\lambda)'\theta + \frac{1}{2}\theta' H(\lambda)\theta \tag{3}$$

where $\theta, b \in \mathcal{R}^s$, $a \in \mathcal{R}$, and $H \in \mathcal{R}^{s \times s}$. For a minimum of (3) to exist requires that H be positive definite. It can be obtained by solving the linear system

$$\frac{\partial C}{\partial \theta} = b + H\theta = 0 \tag{4}$$

which yields the solution

$$\theta(\lambda) = -H^{-1}(\lambda)b(\lambda). \tag{5}$$

Assuming that E only depends on λ through θ, the gradient of the model selection criterion E with respect to λ is

$$\frac{\partial E}{\partial \lambda} = \frac{\partial E}{\partial \theta}\frac{\partial \theta}{\partial \lambda}.$$

If there were a direct dependency (not through θ), an extra partial derivative would have to be added.
For example, in the case of the cross-validation criteria,

$$\frac{\partial E_{cv}}{\partial \theta} = \frac{1}{K}\sum_i \frac{1}{|S_2^i|}\sum_{z_t \in S_2^i} \frac{\partial Q(\theta, z_t)}{\partial \theta}.$$

In the quadratic case, the influence of λ on θ is spelled out by (5), yielding

$$\frac{\partial \theta_i}{\partial \lambda} = -\sum_j \frac{\partial H^{-1}_{i,j}}{\partial \lambda} b_j - \sum_j H^{-1}_{i,j}\frac{\partial b_j}{\partial \lambda} \tag{6}$$

Although the second sum can be readily computed, $\frac{\partial H^{-1}_{i,j}}{\partial \lambda}$ in the first sum is more challenging: we consider several methods below. One solution is based on the computation of gradients through the inverse of a matrix. This general but inefficient solution is the following:

$$\frac{\partial H^{-1}_{i,j}}{\partial \lambda} = \sum_{k,l} \frac{\partial H^{-1}_{i,j}}{\partial H_{k,l}}\frac{\partial H_{k,l}}{\partial \lambda}$$

where

$$\frac{\partial H^{-1}_{i,j}}{\partial H_{k,l}} = -H^{-1}_{i,j} H^{-1}_{l,k} + I_{i \neq l, j \neq k} H^{-1}_{i,j} minor(H, j, i)^{-1}_{l',k'}, \tag{7}$$

where $minor(H, j, i)$ denotes the "minor matrix", obtained by removing the j-th row and the i-th column from H, and the indices (l', k') in the above equation refer to the position within a minor matrix that corresponds to the position (l, k) in H (note $l \neq i$ and $k \neq j$). Unfortunately, the computation of this gradient requires $O(s^5)$ multiply-add operations for an $s \times s$ matrix H, which is much more than is required by the inversion of H ($\mathrm{O}(s^3)$ operations). A better solution is based on the following equality: $HH^{-1} = I$, where I is the $s \times s$ identity matrix. This implies, by differentiating with respect to λ: $\frac{\partial H}{\partial \lambda} H^{-1} + H \frac{\partial H^{-1}}{\partial \lambda} = 0$. Isolating $\frac{\partial H^{-1}}{\partial \lambda}$, we get

$$\frac{\partial H^{-1}}{\partial \lambda} = -H^{-1} \frac{\partial H}{\partial \lambda} H^{-1} \tag{8}$$

which requires only about $2s^3$ multiply-add operations.

PLACE FIGURE 1 AROUND HERE

Figure 1: Illustration of forward paths (full lines) and gradient paths (dashed) for computation the model selection criterion E and its derivative with respect to the hyper-parameters (λ), when using the method based on the Cholesky decomposition and back-substitution to solve for the parameters (θ).

An even better solution (which was suggested by Léon Bottou) is to return to equation (4), which can be solved using about $s^3/3$ multiply-add operations (when $\theta \in \mathcal{R}^s$). The idea is to *back-propagate* gradients through each of the operations performed to solve the linear system. The objective is to compute the gradient of

E with respect to H and b through the effect of H and b on θ, in order to finally compute $\frac{\partial E}{\partial \lambda}$, as illustrated in figure 1. The back-propagation costs the same as the linear system solution, i.e., about $s^3/3$ multiply/add operations, so this is the approach that we have kept for our implementation. Since H is the Hessian matrix, it is positive definite and symmetric, and (4) can be solved through the Cholesky decomposition of H (assuming H is full rank, which is likely if the hyper-parameters provide some sort of weight decay). The Cholesky decomposition of a symmetric positive definite matrix H gives $H = LL'$ where L is a lower diagonal matrix (with zeros above the diagonal). It is computed in time $O(s^3)$ as follows:

`for` $i = 1, \ldots, s$

$L_{i,i} = \sqrt{H_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2}$

for $j = i + 1, \ldots, s$

$L_{j,i} = (H_{i,j} - \sum_{k=1}^{i-1} L_{i,k} L_{j,k})/L_{i,i}$

Once the Cholesky decomposition is achieved, the linear system $LL'\theta = -b$ can be easily solved, in two back-substitution steps: first solve $Lu = -b$ for u, then solve $L'\theta = u$ for θ. First step, iterating once forward through the rows of L:

`for` $i = 1, \ldots, s$,

$u_i = (-b_i - \sum_{k=1}^{i-1} L_{i,k} u_k)/L_{i,i}$.

Second step, iterating once backward through the rows of L:

`for` $i = s, \ldots, 1$,

$\theta_i = (u_i - \sum_{k=i+1}^{s} L_{k,i} \theta_k)/L_{i,i}$.

The computation of the gradient of θ with respect to the elements of H and b proceed in exactly the reverse order. We start by back-propagating through the back-substitution steps, and then through the Cholesky decomposition. Together, the three boxed algorithms that follow allow to compute $\frac{\partial E}{\partial b}$ and $\frac{\partial E}{\partial H}$, starting from the "partial" parameter gradient $\left.\frac{\partial E}{\partial \theta_i}\right|_{\theta_1, \ldots, \theta_s}$ (not taking into account the dependencies of θ_i on θ_j for $j > i$). As intermediate results, the algorithm computes the partial derivatives with respect to u and L, as well as the "full gradients" with respect to θ, $\left.\frac{\partial E}{\partial \theta_i}\right|_{\theta_i}$, taking into account all the dependencies between the θ_i's brought by the recursive computation of the θ_i's.

First back-propagate through the solution of $L'\theta = u$:

```
initialize dEdtheta ← ∂E/∂θ |_{θ_1,...,θ_s}
initialize dEdL ← 0
for i = 1, ..., s
        dEdu_i ← dEdtheta_i / L_{i,i}
        dEdL_{i,i} ← dEdL_{i,i} − dEdtheta_i θ_i / L_{i,i}
        for k = i + 1 ... s
                dEdtheta_k ← dEdtheta_k − dEdtheta_i L_{k,i} / L_{i,i}
                dEdL_{k,i} ← dEdL_{k,i} − dEdtheta_i θ_k / L_{i,i}
```

Then back-propagate through the solution of $Lu = -b$:

```
for i = s, ..., 1
        ∂E/∂b_i ← −dEdu_i / L_{i,i}
        dEdL_{i,i} ← dEdL_{i,i} − dEdu_i u_i / L_{i,i}
        for k = 1, ..., i − 1
                dEdu_k ← dEdu_k − dEdu_i L_{i,k} / L_{i,i}
                dEdL_{i,k} ← dEdL_{i,k} − dEdu_i u_k / L_{i,i}
```

The above algorithm gives us the gradient of the model selection criterion E with respect to coefficient $b(\lambda)$ of the training criterion, as well as with respect to the lower diagonal matrix L, $\texttt{dEdL}_{i,j} = \frac{\partial E}{\partial L_{i,j}}$.

Finally, we back-propagate through the Cholesky decomposition, to convert the gradients with respect to L into gradients with respect to the Hessian $H(\lambda)$:

```
for i = s, ..., 1
        for j = s, ..., i + 1
                dEdL_{i,i} ← dEdL_{i,i} − dEdL_{j,i} L_{j,i} / L_{i,i}
                ∂E/∂H_{i,j} ← dEdL_{j,i} / L_{i,i}
                for k = 1, ..., i − 1
                        dEdL_{i,k} ← dEdL_{i,k} − dEdL_{j,i} L_{j,k} / L_{i,i}
                        dEdL_{j,k} ← dEdL_{j,k} − dEdL_{j,i} L_{i,k} / L_{i,i}
        ∂E/∂H_{i,i} ← (1/2) dEdL_{i,i} / L_{i,i}
        for k = 1, ..., i − 1
                dEdL_{i,k} ← dEdL_{i,k} − dEdL_{i,i} L_{i,k} / L_{i,i}
```

Note that we have only computed gradients with respect to the diagonal and upper diagonal of H because H is symmetric. Once we have the gradients of E with respect to b and H, we use the functional form of $b(\lambda)$ and $H(\lambda)$ to compute the gradient of E with respect to λ:

$$\frac{\partial E}{\partial \lambda} = \sum_i \frac{\partial E}{\partial b_i} \frac{\partial b_i}{\partial \lambda} + \sum_{i,j} \frac{\partial E}{\partial H_{i,j}} \frac{\partial H_{i,j}}{\partial \lambda}$$

(again we assumed that there is no direct dependency from λ to E, otherwise an extra term must be added).

Using this approach rather than the one described in the previous subsection, the overall computation of gradients is therefore in about $s^3/3$ multiply-add operations rather than $O(s^5)$. The most expensive step is the back-propagation through the Cholesky decomposition itself (three nested s-iterations loops). Note that this step may be shared if there are several linear systems to solve with the same Hessian matrix. For example this will occur in linear regression with multiple outputs because H is block-diagonal, with one block for each set of parameters associated to one output, and all blocks being the same (equal to the input "design matrix" $\sum_t x_t x_t'$, denoting x_t the input training vectors). Only a single Cholesky computation needs to be done, shared across all blocks.

3.1 Weight Decays for Linear Regression

In this subsection, we illustrate the method in the particular case of multiple weight decays for linear regression, with K-fold cross-validation as the model selection criterion. The hyper-parameter λ_j will be a weight decay associated to the j-th input variable. The training criterion for the k-th partition is

$$C_k = \frac{1}{|S_1^k|} \sum_{(x_t,y_t)\in S_1^k} \frac{1}{2}(\Theta x_t - y_t)'(\Theta x_t - y_t) + \frac{1}{2}\sum_j \lambda_j \sum_i \Theta_{i,j}^2 \tag{9}$$

The objective is to penalize separately each of the input variables (as in (MacKay and Neal, 1994; Neal, 1998)), a kind of "soft variable selection" (see (Latendresse and Bengio, 1999) for more discussion and experiments with this setup).

The training criterion is quadratic, as in (3), with coefficients

$$a = \frac{1}{2}\sum_t y_t' y_t, \qquad b_{(ij)} = -\sum_t y_{t,i} x_{t,j}, \qquad H_{(ij),(i'j')} = \delta_{i,i'} \sum_t x_{t,j} x_{t,j'} + \delta_{i,i'}\delta_{j,j'}\lambda_j,$$

where $\delta_{i,j} = 1$ when $i = j$ and 0 otherwise, and (ij) is an index corresponding to indices (i, j) in the weight matrix Θ, e.g., $(ij) = (i-1) \times s + j$. From the above definition of the coefficients of C, we obtain their partial derivatives with respect to λ:

$$\frac{\partial b}{\partial \lambda} = 0, \qquad \frac{\partial H_{(ij),(i'j')}}{\partial \lambda_k} = \delta_{i,i'}\delta_{j,j'}\delta_{j,k}.$$

Plugging the above definitions of the coefficients and their derivatives in the equations and algorithms of the previous subsection, we have therefore obtained an algorithm for computing the gradient of the model selection criterion with respect to the input weight decays of a linear regression.

Note that here H is block-diagonal, with m identical blocks of size $(n+1)$, so the Cholesky decomposition (and similarly back-propagating through it) can be performed in about $(s/m)^3/3$ multiply-add operations rather than $s^3/3$ operations, where m is the number of outputs (the dimension of the output variable).

4 Non-Quadratic Criterion: Hyper-Parameters Gradient

If the training criterion C is not quadratic in terms of the parameters θ, it will in general be necessary to apply an iterative numerical optimization algorithm to minimize the training criterion. In this section we consider what happens after this minimization is performed, i.e., at a value of θ where $\frac{\partial C}{\partial \theta}$ is approximately zero and $\frac{\partial^2 C}{\partial \theta^2}$ is positive definite (otherwise we would not be at a minimum of C). The minimization of $C(\theta, \lambda, D)$ defines a function $\theta(\lambda, D)$ (equation 1). With our assumption of smoothness of C, the implicit function theorem tells us that this function exists locally and is differentiable. To obtain this function we write

$$F(\theta, \lambda) = \frac{\partial C}{\partial \theta} = 0,$$

evaluated at $\theta = \theta(\lambda, D)$ (at a minimum of C). Differentiating the above equation with respect to λ, we obtain

$$\frac{\partial F}{\partial \theta}\frac{\partial \theta}{\partial \lambda} + \frac{\partial F}{\partial \lambda} = 0$$

so we obtain a **general formula for the gradient of the fitted parameters with respect to the hyper-parameters**:

$$\boxed{\frac{\partial \theta(\lambda, D)}{\partial \lambda} = -(\frac{\partial^2 C}{\partial \theta^2})^{-1}\frac{\partial^2 C}{\partial \lambda \partial \theta} = -H^{-1}\frac{\partial^2 C}{\partial \lambda \partial \theta}} \quad (10)$$

Let us see how this result relates to the special case of a quadratic training criterion, $C = a + b'\theta + \frac{1}{2}\theta' H \theta$:

$$\frac{\partial \theta}{\partial \lambda} = -H^{-1}(\frac{\partial b}{\partial \lambda} + \frac{\partial H}{\partial \lambda}\theta) = -H^{-1}\frac{\partial b}{\partial \lambda} + H^{-1}\frac{\partial H}{\partial \lambda}H^{-1}b$$

where we have substituted $\theta = -H^{-1}b$. Using the equality (8), we obtain the same formula as in eq. (6).

Let us consider more closely the case of a neural network with one layer of hidden units with hyperbolic tangent activations, a linear output layer, squared loss, and hidden layer weights $W_{i,j}$. For example, if we want to use hyper-parameters for penalizing the use of inputs, we have a criterion similar to (9),

$$C_k = \frac{1}{|S_1^k|} \sum_{(x_t,y_t)\in S_1^k} \frac{1}{2}(f_\theta(x_t) - y_t)'(f_\theta(x_t) - y_t) + \frac{1}{2}\sum_j \lambda_j \sum_i W_{i,j}^2.$$

with $C = \sum_k C_k$, and the cross-derivatives are easy to compute:

$$\frac{\partial^2 C}{\partial W_{i,j} \partial \lambda_k} = \delta_{k,j} W_{i,j}.$$

The Hessian and its inverse require more work, but can be done respectively in at most $O(s^2)$ and $O(s^3)$ operations. See for example (Bishop, 1992) for the exact computation of the Hessian for multi-layer neural networks. See (Becker and LeCun, 1989; LeCun, Denker and Solla, 1990) for a diagonal approximation which can be computed and inverted in $O(s)$ operations.

5 Summary of Experiments and Conclusions

In this paper, we have presented a new methodology for simultaneously optimizing several hyper-parameters, based on the computation of the gradient of a model selection criterion with respect to the hyper-parameters, taking into account the influence of the hyper-parameters on the parameters. We have considered both the simpler case of a training criterion that is quadratic with respect to the parameters ($\theta \in \mathcal{R}^s$) and the more general non-quadratic case. We have shown a particularly efficient procedure in the quadratic case that is based on back-propagating gradients through the Cholesky decomposition and back-substitutions. This was an improvement: we have arrived at this $s^3/3$-operations procedure after studying first an $O(s^5)$ procedure and then a $2s^3$-operations procedure. In the particular case of input weight decays for linear regression, the computation can even be reduced to about $(s/m)^3/3$ operations when there are m outputs.

We have performed preliminary experiments with the proposed methodology in several simple cases, using conjugate gradients to optimize the hyper-parameters.

The application to linear regression with weight decays for each input is described in (Latendresse and Bengio, 1999). The hyper-parameter optimization algorithm is used to perform a soft selection of the input variables. A large weight decay on one of the inputs effectively forces the corresponding weights to very small values. Comparisons on simulated data sets are made in (Latendresse and Bengio, 1999) with ordinary regression as well as with stepwise regression methods and the adaptive ridge (Grandvalet, 1998) or LASSO (Tibshirani, 1995).

Another type of application of the proposed method has been explored, in the context of a "real-world" problem of non-stationary time-series prediction (Bengio and Dugas, 1999). In this case, an extension of the cross-validation criterion to sequential data which may be non-stationary is used. Because of this non-stationarity, recent data may sometimes be more relevant to current predictions than older data. The training criterion is a sum of weighted errors for the past examples, and these weights are given by a parametrized function of time (as the $w_i(\lambda)$ in eq. 2). The parameters of that function are two hyper-parameters that control when a transition in the unknown generating process would have occurred and how strong that change was or should be trusted. In these experiments, the weight given to past data points is a sigmoid function of the time: the threshold and the slope of the sigmoid are the hyper-parameters, representing respectively the time of a strong transition and the strength of that transition. Optimizing these hyper-parameters, we obtained statistically significant improvements in predicting one-month ahead future volatility of Canadian stocks. The comparisons were made against several linear, constant, and ARMA models of the volatility. The experiments were performed on monthly return data from 473 Canadian stocks from 1976 to 1996. The measure of performance is the average out-of-sample squared error in predicting the squared returns. Single-sided significance tests were performed taking into account the auto-covariance in the temporal series of errors and the covariance of the errors between the compared models. When comparing the prediction of the first moment (expected return), no model significantly improved on the historical average of stock returns (constant

model). When comparing the prediction of the second moment (expected squared returns), the method based on hyper-parameters optimization beat all the other methods, with a p-value of 1% or less.

What remains to be done? first, more experiments, in particular with the non-quadratic case (e.g., MLPs), and with model selection criteria other than cross-validation (which has large variance (Breiman, 1996)). Second, there are important theoretical questions that remain unanswered concerning the amount of over-fitting that can be brought when too many hyper-parameters are optimized. As we have outlined in the introduction, the situation with hyper-parameters may be compared with the situation of parameters. However, whereas the form of the training criterion as a sum of independent errors allows to define the capacity for a class of functions and relate it to the difference between generalization error and training error, it does not appear clearly to us how a similar analysis could be performed for hyper-parameters.

Acknowledgements

The author would like to thank Léon Bottou, Pascal Vincent, François Blanchette, and François Gingras, as well as the NSERC Canadian funding agency.

References

Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–728.

Becker, S. and LeCun, Y. (1989). Improving the convergence of back-propagation learning with second order methods. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37, Pittsburg 1988. Morgan Kaufmann, San Mateo.

Bengio, Y. and Dugas, C. (1999). Learning simple non-stationarities with hyper-parameters. *submitted to Machine Learning*.

Bishop, C. (1992). Exact calculation of the Hessian matrix for the multi-layer perceptron. *Neural Computation*, 4(4):494–501.

Bottou, L. (1998). Online algorithms and stochastic approximations. In Saad, D., editor, *Online Learning in Neural Networks*. Cambridge University Press, to appear, Cambridge, UK.

Breiman, L. (1996). Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24 (6):2350–2383.

Craven, P. and Wahba, G. (1979). Smoothing noisy data with spline functions. *Numerical Mathematics*, 31:377–403.

Grandvalet, Y. (1998). Least absolute shrinkage is equivalent to quadratic penalization. In Niklasson, L., Boden, M., and Ziemske, T., editors, *ICANN'98*, volume 1 of *Perspectives in Neural Computing*, pages 201–206. Springer.

Guyon, I., Vapnik, V., Boser, B., Bottou, L., and la, S. S. (1992). Structural risk minimization for character recognition. In Moody, J., Hanson, S., and Lipmann, R., editors, *Advances in Neural Information Processing Systems 4*, pages 471–479, San Mateo CA. Morgan Kaufmann.

Hinton, G. (1987). Learning translation invariant in massively parallel networks. In de Bakker, J., Nijman, A., and Treleaven, P., editors, *Proceedings of PARLE Conference on Parallel Architectures and Languages Europe*, pages 1–13, Berlin. Springer-Verlag.

Hoerl, A. and Kennard, R. (1970). Ridge regression: biased estimation for non-orthogonal problems. *Technometrics*, 12:55–67.

Latendresse, S. and Bengio, Y. (1999). Linear regression and the optimization of hyper-parameters. *submitted to NIPS'99*.

LeCun, Y., Denker, J., and Solla, S. (1990). Optimal brain damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 598–605, Denver, CO. Morgan Kaufmann, San Mateo.

MacKay, D. and Neal, R. (1994). Automatic relevance determination. Unpublished report. See also MacKay D., 1995, Probable Neworks and Plausible Predictions – A Review of Practical Bayesian Methods for Supervised Neural Networks, in *Neutwork: Computation in Neural Systems*, v. 6, pp. 469–505.

Neal, R. (1998). Assessing relevance determination methods using delve. In Bishop, C., editor, *Neural Networks and Machine Learning*, pages 97–129. Springer-Verlag.

Poggio, T., Torre, V., and Koch, C. (1985). Computational vision and regularization theory. *Nature*, 317(26):314–319.

Rissanen, J. (1990). *Stochastic Complexity in Statistical Inquiry.* World Scientific, Singapore.

Tibshirani, R. (1995). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58:267–288.

Tikhonov, A. and Arsenin, V. (1977). *Solutions of Ill-posed Problems.* W.H. Winston, Washington D.C.

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

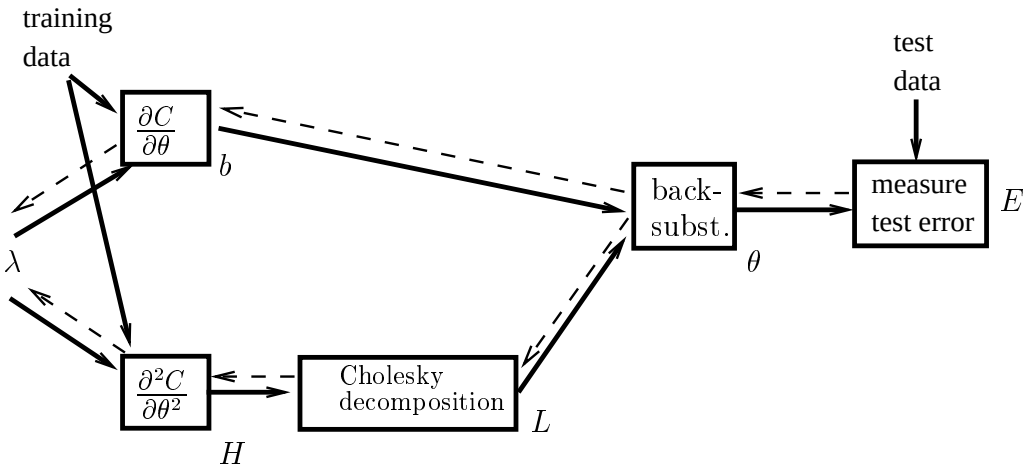

FIGURE 1 FROM BENGIO'S PAPER # 2045