

Gradient-Based Optimization of Hyperparameters

Yoshua Bengio

*Département d'informatique et recherche opérationnelle, Université de Montréal,
Montréal, Québec, Canada, H3C 3J7*

Many machine learning algorithms can be formulated as the minimization of a training criterion that involves a hyperparameter. This hyperparameter is usually chosen by trial and error with a model selection criterion. In this article we present a methodology to optimize several hyperparameters, based on the computation of the gradient of a model selection criterion with respect to the hyperparameters. In the case of a quadratic training criterion, the gradient of the selection criterion with respect to the hyperparameters is efficiently computed by backpropagating 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 hyperparameter 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 hyperparameter, kept fixed during this minimization. For example, in the regularization framework (Tikhonov & Arsenin, 1977; Poggio, Torre, & Koch, 1985), one hyperparameter 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 in that case as ridge regression; Hoerl & Kennard, 1970), to penalize the L2-norm of the parameters. Increasing the penalty term (increasing the weight decay hyperparameter) corresponds to reducing the *effective capacity* (Guyon, Vapnik, Boser, Bottou, & Solla, 1992) by forcing the solution to be in a zero-centered hypersphere 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 hyperparameters, more generally to compare and choose among models that may have a different capacity. Many model selection criteria have been proposed in the past

(Vapnik, 1982; Akaike, 1974; Craven & Wahba, 1979). When there is only a single hyperparameter, one can easily explore how its value affects the model selection criterion: typically one tries a finite number of values of the hyperparameter and picks the one that gives the lowest value of the model selection criterion.

In this article, we present a methodology to select simultaneously many hyperparameters using the gradient of the model selection criterion with respect to the hyperparameters. This methodology can be applied when some differentiability and continuity conditions of the training criterion are satisfied. The use of multiple hyperparameters has already been proposed in the Bayesian literature. One hyperparameter per input feature was used to control the prior on the parameters associated with that input feature (MacKay, 1995; Neal, 1998). In this case, the hyperparameters can be interpreted as scale parameters for the prior distribution on the parameters, for different directions in parameter space. Note that independently of this work, Larsen et al. (1998) have proposed a procedure similar to the one presented here, for optimizing regularization parameters of a neural network (different weight decays for different layers of the network). In sections 2, 3, and 4, we explain how the gradient with respect to the hyperparameters can be computed. In the conclusion, we briefly describe the results of preliminary experiments performed with the proposed methodology (described in more detail in Latendresse, 1999, and Bengio & Dugas, 1999), and we raise some important open questions concerning the kind of “overfitting” that can occur with the proposed methodology.

2 Objective Functions for Hyperparameters

We are given a set of independent data points, $D = \{z_1, \dots, z_T\}$, 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 with 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 hyperparameters.

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 multilayer 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

hyperparameters λ :

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

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

$$\theta(\lambda, D) = \operatorname{argmin}_{\theta} C(\theta, \lambda, D). \tag{2.1}$$

An example of a training criterion with hyperparameters is

$$C = \sum_{(x_i, y_i) \in D} w_i(\lambda) (f_{\theta}(x_i) - y_i)^2 + \theta' A(\lambda) \theta, \tag{2.2}$$

where the hyperparameters 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 & Dugas, 1999; Latendresse, 1999).

2.2 Model Selection Criteria. The model selection criterion E is used to select hyperparameters or more generally to choose one model among several. Ideally, it should be the expected generalization error (for a fixed λ), but $P(Z)$ is unknown, so many alternatives—approximations, bounds, or empirical estimates—have been proposed. Most model selection criteria have been proposed for selecting a single hyperparameter that controls the “complexity” of the class of functions in which the learning algorithm finds a solution, such as 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 & Wahba, 1979). Other 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 , $S_1^1 \cup S_2^1, S_1^2 \cup S_2^2, \dots, S_1^K \cup S_2^K$:

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

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. And 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 overfitting the hyperparameters. In this sense, the use of hyperparameters proposed in this article can be very different from the common use in which a hyperparameter helps to control overfitting. Instead, a blind use of the extra freedom brought by many hyperparameters could deteriorate generalization.

3 Optimizing Hyperparameters 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 hyperparameters 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):

$$C = a(\lambda) + b(\lambda)' \theta + \frac{1}{2} \theta' H(\lambda) \theta, \quad (3.1)$$

where $\theta, b \in \mathcal{R}^s$, $a \in \mathcal{R}$, and $H \in \mathcal{R}^{s \times s}$. For a minimum of equation 3.1 to exist requires that H be positive definite. It can be obtained by solving the linear system

$$\frac{\partial C}{\partial \theta} = b + H\theta = 0, \quad (3.2)$$

which yields the solution

$$\theta(\lambda) = -H^{-1}(\lambda)b(\lambda). \quad (3.3)$$

Assuming that E depends only 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 equation 3.3, yielding

$$\frac{\partial \theta_i}{\partial \lambda} = - \sum_j \frac{\partial H_{i,j}^{-1}}{\partial \lambda} b_j - \sum_j H_{i,j}^{-1} \frac{\partial b_j}{\partial \lambda}. \quad (3.4)$$

Although the second sum can be readily computed, $\frac{\partial H_{ij}^{-1}}{\partial \lambda}$ 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:

$$\frac{\partial H_{ij}^{-1}}{\partial \lambda} = \sum_{k,l} \frac{\partial H_{ij}^{-1}}{\partial H_{k,l}} \frac{\partial H_{k,l}}{\partial \lambda},$$

where

$$\frac{\partial H_{ij}^{-1}}{\partial H_{k,l}} = -H_{ij}^{-1} H_{l,k}^{-1} + I_{i \neq l, j \neq k} H_{ij}^{-1} \text{minor}(H, j, i)_{l,k}^{-1}, \tag{3.5}$$

where $\text{minor}(H, j, i)$ denotes the “minor matrix,” obtained by removing the j th row and the i th column from H , and the indices (l, k) in equation 3.5 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 ($O(s^3)$ 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

$$\frac{\partial H^{-1}}{\partial \lambda} = -H^{-1} \frac{\partial H}{\partial \lambda} H^{-1}, \tag{3.6}$$

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

An even better solution (which was suggested by Léon Bottou) is to return to equation 3.2, which can be solved using about $s^3/3$ multiply-add operations (when $\theta \in \mathcal{R}^s$). The idea is to backpropagate 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 compute $\frac{\partial E}{\partial \lambda}$, as illustrated in Figure 1. The backpropagation costs the same as the linear system solution—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 equation 3.2 can be solved through the Cholesky decomposition of H (assuming H is full rank, which is likely if the hyperparameters 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:

$$\begin{aligned} &\text{for } i = 1, \dots, s \\ &L_{i,i} = \sqrt{H_{i,i} - \sum_{k=1}^{i-1} L_{i,k}^2} \\ &\text{for } j = i + 1, \dots, s \\ &L_{j,i} = (H_{i,j} - \sum_{k=1}^{i-1} L_{i,k} L_{j,k}) / L_{i,i}. \end{aligned}$$

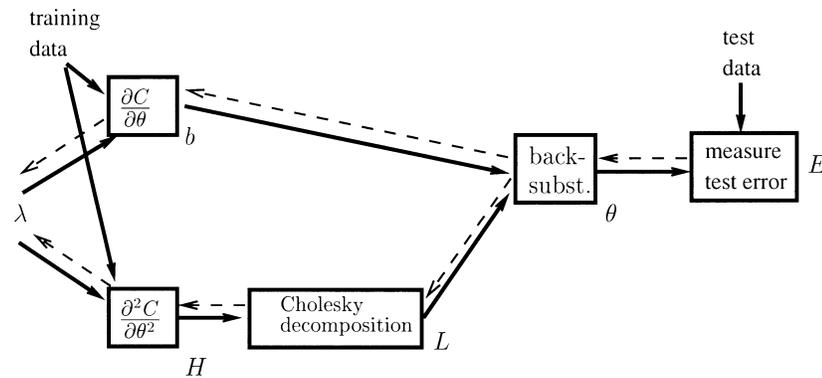


Figure 1: Illustration of forward paths (full lines) and gradient paths (dashed) for computation of the model selection criterion E and its derivative with respect to the hyperparameters (λ), when using the method based on the Cholesky decomposition and backsubstitution to solve for the parameters (θ).

Once the Cholesky decomposition is achieved, the linear system $LL'\theta = -b$ can be easily solved in two backsubstitution steps: (1) first solve $Lu = -b$ for u ; (2) solve $L'\theta = u$ for θ :

1. Iterate once forward through the rows of L :

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

2. Iterate once backward through the rows of L :

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

The computation of the gradient of θ with respect to the elements of H and b proceeds in exactly the reverse order. We start by backpropagating through the backsubstitution steps, and then through the Cholesky decomposition. Together, the three algorithms that follow allow computing $\frac{\partial E}{\partial b}$ and $\frac{\partial E}{\partial H}$, starting from the “partial” parameter gradient $\frac{\partial E}{\partial \theta_i} \Big|_{\theta_1, \dots, \theta_s}$ (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 , $\frac{\partial E}{\partial \theta_i} \Big|_{\theta_i}$, taking into account all the dependencies between the θ_i ’s brought by the recursive computation of the θ_i ’s.

First backpropagate through the solution of $L'\theta = u$:

```
initialize dEdtheta ←  $\frac{\partial E}{\partial \theta} \Big|_{\theta_1, \dots, \theta_s}$ 
initialize dEdL ← 0
```

```

for  $i = 1, \dots, s$ 
   $dE_{du_i} \leftarrow dEd_{\theta_i} / L_{i,i}$ 
   $dEd_{L_{i,i}} \leftarrow dEd_{L_{i,i}} - dEd_{\theta_i} \theta_i / L_{i,i}$ 
  for  $k = i + 1 \dots s$ 
     $dEd_{\theta_k} \leftarrow dEd_{\theta_k} - dEd_{\theta_i} L_{k,i} / L_{i,i}$ 
     $dEd_{L_{k,i}} \leftarrow dEd_{L_{k,i}} - dEd_{\theta_i} \theta_k / L_{i,i}$ 

```

Then backpropagate through the solution of $Lu = -b$:

```

for  $i = s, \dots, 1$ 
   $\frac{\partial E}{\partial b_i} \leftarrow -dE_{du_i} / L_{i,i}$ 
   $dEd_{L_{i,i}} \leftarrow dEd_{L_{i,i}} - dE_{du_i} u_i / L_{i,i}$ 
  for  $k = 1, \dots, i - 1$ 
     $dE_{du_k} \leftarrow dE_{du_k} - dE_{du_i} L_{i,k} / L_{i,i}$ 
     $dEd_{L_{i,k}} \leftarrow dEd_{L_{i,k}} - dE_{du_i} u_k / L_{i,i}$ 

```

The above algorithm gives 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 , $dEd_{L_{i,j}} = \frac{\partial E}{\partial L_{i,j}}$.

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

```

for  $i = s, \dots, 1$ 
  for  $j = s, \dots, i + 1$ 
     $dEd_{L_{i,i}} \leftarrow dEd_{L_{i,i}} - dEd_{L_{j,i}} L_{j,i} / L_{i,i}$ 
     $\frac{\partial E}{\partial H_{i,j}} \leftarrow dEd_{L_{j,i}} / L_{i,i}$ 
    for  $k = 1, \dots, i - 1$ 
       $dEd_{L_{i,k}} \leftarrow dEd_{L_{i,k}} - dEd_{L_{j,i}} L_{j,k} / L_{i,i}$ 
       $dEd_{L_{j,k}} \leftarrow dEd_{L_{j,k}} - dEd_{L_{j,i}} L_{i,k} / L_{i,i}$ 
   $\frac{\partial E}{\partial H_{i,i}} \leftarrow \frac{1}{2} dEd_{L_{i,i}} / L_{i,i}$ 
  for  $k = 1, \dots, i - 1$ 
     $dEd_{L_{i,k}} \leftarrow dEd_{L_{i,k}} - dEd_{L_{i,i}} L_{i,k} / L_{i,i}$ 

```

Note that we have computed gradients only 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 backpropagation through the Cholesky decomposition itself (three nested s -iterations loops). 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 section, 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 hyperparameter λ_j will be a weight decay associated with the j th input variable. The training criterion for the k th partition is

$$C_k = \frac{1}{|S_1^k|} \sum_{(x_t, y_t) \in S_1^k} \frac{1}{2} (\Theta x_t - y_t)' (\Theta x_t - y_t) + \frac{1}{2} \sum_j \lambda_j \sum_i \Theta_{i,j}^2. \quad (3.7)$$

The objective is to penalize separately each of the input variables (as in MacKay, 1995; Neal, 1998), a kind of “soft variable selection” (see Latendresse, 1999, for more discussion and experiments with this setup). The training criterion is quadratic, as in equation 3.1, with coefficients

$$a = \frac{1}{2} \sum_t y_t y_t, \quad b_{(ij)} = - \sum_t y_t x_{t,j},$$

$$H_{(ij), (i'j')} = \delta_{i,i'} \sum_t x_{t,j} x_{t,j'} + \delta_{i,i'} \delta_{j,j'} \lambda_j,$$

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 , for example, $(ij) = (i-1) \times s + j$. From the above definition of the coefficients of C , we obtain their partial derivatives with respect to λ :

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

By plugging the above definitions of the coefficients and their derivatives in the equations and algorithms of the previous section, 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 backpropagating 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 Nonquadratic Criterion: Hyperparameters 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—at a value of θ where $\partial C/\partial\theta$ is approximately zero and $\partial^2 C/\partial\theta^2$ is positive definite (otherwise we would not be at a minimum of C). The minimization of $C(\theta, \lambda, D)$ defines a function $\theta(\lambda, D)$ (see equation 2.1). 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 hyperparameters:

$$\frac{\partial \theta(\lambda, D)}{\partial \lambda} = - \left(\frac{\partial^2 C}{\partial \theta^2} \right)^{-1} \frac{\partial^2 C}{\partial \lambda \partial \theta} = -H^{-1} \frac{\partial^2 C}{\partial \lambda \partial \theta}. \tag{4.1}$$

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

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

where we have substituted $\theta = -H^{-1}b$. Using the equality 3.6, we obtain the same formula as in equation 3.4.

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 hyperparameters for penalizing the use of inputs, we have a criterion similar to equation 3.7:

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

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 multilayer neural networks. See Becker and LeCun (1989) and 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

We have presented a new methodology for simultaneously optimizing several hyperparameters, based on the computation of the gradient of a model selection criterion with respect to the hyperparameters, taking into account the influence of the hyperparameters on the parameters. We have considered the simpler case of a training criterion that is quadratic with respect to the parameters ($\theta \in \mathcal{R}^s$) and the more general nonquadratic case. We have shown a particularly efficient procedure in the quadratic case that is based on backpropagating gradients through the Cholesky decomposition and backsubstitutions. 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 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 hyperparameters. The application to linear regression with weight decays for each input is described in Latendresse (1999). The hyperparameter 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 (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 nonstationary time-series prediction (Bengio & Dugas, 1999). In this case, an extension of the cross-validation criterion to sequential data that may be nonstationary is used. Because of this nonstationarity, 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 parameterized function of time (as the $w_i(\lambda)$ in equation 2.2). The parameters of that function are two hyperparameters 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 hyperparameters, representing, respectively, the time of a strong transition and the strength of that transition. Optimizing

these hyperparameters, 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 hyperparameters optimization beats all the other methods, with a p -value of 1% or less.

What remains to be done? First, we need more experiments, in particular with the nonquadratic 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 overfitting that can be brought when too many hyperparameters are optimized. As we outlined in the introduction, the situation with hyperparameters may be compared with the situation of parameters. However, whereas the form of the training criterion as a sum of independent errors allows defining the capacity for a class of functions and relating it to the difference between generalization error and training error, more work needs to be done to obtain a similar analysis for hyperparameters.

Acknowledgments

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*, 716–728.
- Becker, S., & LeCun, Y. (1989). Improving the convergence of back-propagation learning with second order methods. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School* (pp. 29–37). San Mateo, CA: Morgan Kaufmann.
- Bengio, Y., & Dugas, C. (1999). *Learning simple non-stationarities with hyperparameters*. Submitted.
- Bishop, C. (1992). Exact calculation of the Hessian matrix for the multi-layer perceptron. *Neural Computation*, *4*, 494–501.

- Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad (Ed.), *Online learning in neural networks*. Cambridge: Cambridge University Press.
- Breiman, L. (1996). Heuristics of instability and stabilization in model selection. *Annals of Statistics*, *24*, 2350–2383.
- Craven, P., & 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 L. Niklasson, M. Boden, & T. Ziemke (Eds.), *ICANN'98* (pp. 201–206). Berlin: Springer-Verlag.
- Guyon, I., Vapnik, V., Boser, B., Bottou, L., & Solla, S. A. (1992). Structural risk minimization for character recognition. In J. Moody, S. Hanson, & R. Lipmann (Eds.), *Advances in neural information processing systems*, *4* (pp. 471–479). San Mateo, CA: Morgan Kaufmann.
- Hinton, G. (1987). Learning translation invariant in massively parallel networks. In J. de Bakker, A. Nijman, & P. Treleaven (Eds.), *Proceedings of PARLE Conference on Parallel Architectures and Languages Europe* (pp. 1–13). Berlin: Springer-Verlag.
- Hoerl, A., & Kennard, R. (1970). Ridge regression: Biased estimation for non-orthogonal problems. *Technometrics*, *12*, 55–67.
- Larsen, J., Svarer, C., Andersen, L. N., & Hansen, L. K. (1998). Adaptive regularization in neural networks modeling. In G. B. Orr & K.-R. Muller (Eds.), *Neural Networks: Tricks of the Trade* (pp. 113–132). Berlin: Springer-Verlag.
- Latendresse, S. (1999). *Utilisation d'hyper-parametres pour la selection de variable*. M.Sc. thesis, University of Montreal.
- LeCun, Y., Denker, J., & Solla, S. (1990). Optimal brain damage. In D. Touretzky (Ed.), *Advances in neural information processing systems*, *2* (pp. 598–605). San Mateo, CA: Morgan Kaufmann.
- MacKay D. (1995). Probable networks and plausible predictions—a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, *6*, 469–505. Available online at: <http://www.iro.umontreal.ca/~lisa>.
- Neal, R. (1998). Assessing relevance determination methods using delve. In C. Bishop (Ed.), *Neural networks and machine learning* (pp. 97–129). Berlin: Springer-Verlag.
- Poggio, T., Torre, V., & Koch, C. (1985). Computational vision and regularization theory. *Nature*, *317*, 314–319.
- Rissanen, J. (1990). *Stochastic complexity in statistical inquiry*. Singapore: World Scientific.
- Tibshirani, R. (1995). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, *58*, 267–288.
- Tikhonov, A., & Arsenin, V. (1977). *Solutions of ill-posed problems*. Washington, DC: W. H. Winston.
- Vapnik, V. (1982). *Estimation of dependences based on empirical data*. Berlin: Springer-Verlag.

Received June 1, 1999; accepted August 30, 1999.