ADAPTIVE REGULARIZATION OF NEURAL NETWORKS USING CONJUGATE GRADIENT

Cyril Goutte and Jan Larsen

CONNECT, Department of Mathematical Modelling, Building 321
Technical University of Denmark, DK-2800 Lyngby, Denmark
emails: cg,jl@imm.dtu.dk
www: http://eivind.imm.dtu.dk

ABSTRACT

Recently we suggested a regularization scheme which iteratively adapts regularization parameters by minimizing validation error using simple gradient descent. In this contribution we present an improved algorithm based on the conjugate gradient technique. Numerical experiments with feed-forward neural networks successfully demonstrate improved generalization ability and lower computational cost.

1. INTRODUCTION

Neural networks are flexible tools for regression, time-series modeling and pattern recognition which find expression in universal approximation theorems [6].

The risk of over-fitting on noisy data is of major concern in neural network design, as exemplified by the bias-variance dilemma, see e.g., [5]. Using regularization serves two purposes: first, it remedies numerical instabilities during training by imposing smoothness on the cost function; secondly, regularization is a tool for reducing variance by introducing extra bias. The overall goal is to minimize the generalization error, i.e., the sum of the bias, the variance, and inherent noise.

In recent publications [1], [10], [11] we proposed an adaptive scheme for tuning the amount of regularization by minimizing an empirical estimate of the generalization error, e.g., the hold-out cross-validation error or K-fold cross-validation error. The adaptive scheme was based on simple gradient descent which is known to have poor convergence properties [15]. Consequently, we suggest an improved scheme based on conjugate gradient minimization[1] [3, 13] of the simple hold-out validation error.

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). CG was supported by a DTU research grant; JL furthermore acknowledges the Radio Parts Foundation for financial support.

[1]Unfortunately, true second order optimization techniques are precluded since they involve 3rd order derivatives of the cost function w.r.t. to network weights.

2. TRAINING AND GENERALIZATION

Suppose the neural network is described by the vector function $\boldsymbol{f}(\boldsymbol{x};\boldsymbol{w})$ where $\boldsymbol{x}$ is the input vector and $\boldsymbol{w}$ is the vector of network weights and thresholds with dimensionality m. The objective is to use the neural network to approximate the conditional input-output distribution $p(\boldsymbol{y}|\boldsymbol{x})$ or its moments. Normally, we model only the conditional expectation $E[\boldsymbol{y}|\boldsymbol{x}]$ which is optimal in a least squares sense.

Assume that we have available a dataset, $\mathcal{D} = \{(\boldsymbol{x}(k), \boldsymbol{y}(k))\}_{k=1}^{N}$, of N input-output examples split into two disjoint sets: a validation set, $\mathcal{V}$, with $N_v = \lceil \gamma N \rceil$ examples[2] for estimation of regularization, and a training set, $\mathcal{T}$, with $N_t = N - N_v$ examples for estimation of network parameters. $0 \le \gamma \le 1$ is referred to as the split-ratio.

The neural network is trained by minimizing a cost function which is the sum of a loss function (or training error), $S_{\mathcal{T}}(\boldsymbol{w})$, and a regularization term $R(\boldsymbol{w}, \boldsymbol{\kappa})$, where $\boldsymbol{\kappa}$ is the set of regularization parameters:

$$\begin{aligned} C(\boldsymbol{w}) &= S_{\mathcal{T}}(\boldsymbol{w}) + R(\boldsymbol{w}, \boldsymbol{\kappa}) \\ &= \frac{1}{N_t} \sum_{k=1}^{N_t} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \boldsymbol{w}\right) + R(\boldsymbol{w}, \boldsymbol{\kappa}) \quad (1) \end{aligned}$$

where $\ell(\cdot)$ measures the cost associated with estimating output $\boldsymbol{y}(k)$ by the network prediction $\widehat{\boldsymbol{y}}(k) = \boldsymbol{f}(\boldsymbol{x}(k); \boldsymbol{w})$. In the experimental section we consider the mean squared error loss $\ell = (\boldsymbol{y} - \widehat{\boldsymbol{y}})^2$. $N_t \equiv |\mathcal{T}|$ defines the number of training examples and k indexes the specific example.

Training provides the estimated weight vector $\widehat{\boldsymbol{w}} = \mathrm{argmin}_{\boldsymbol{w}}\, C(\boldsymbol{w})$. The validation set consists of another $N_v \equiv |\mathcal{V}|$ examples and the validation error of the

[2]$\lceil \cdot \rceil$ denotes rounding upwards to the nearest integer.

trained network reads

$$S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) = \frac{1}{N_v}\sum_{k=1}^{N_v} \ell\left(\boldsymbol{y}(k), \widehat{\boldsymbol{y}}(k); \widehat{\boldsymbol{w}}\right) \tag{2}$$

where the sum runs over the N_v validation examples. $S_{\mathcal{V}}(\widehat{\boldsymbol{w}})$ is thus an unbiased estimate of the generalization error defined as $G(\widehat{\boldsymbol{w}}) = E_{\boldsymbol{x},\boldsymbol{y}}\{\ell(\boldsymbol{y}, \widehat{\boldsymbol{y}}; \widehat{\boldsymbol{w}})\}$, i.e., the expectation of the loss function w.r.t. to the (unknown) joint input-output distribution.

Ideally we need N_v as large as possible which leaves only few data for training, thus increasing the true generalization error $G(\widehat{\boldsymbol{w}})$. Consequently there exists an optimal split-ratio γ corresponding to a trade-off between the conflicting aims, see e.g., [8], [9].

A minimal necessary requirement for a procedure which estimates the network parameters on the training set and optimizes the amount of regularization from a validation set is: the generalization error of the regularized network should be smaller than that of the unregularized network trained on the full data set $\mathcal{D}$. However, this is not always the case (see e.g., [11]), and is indeed the quintessence of the so-called "no free lunch" theorems.

3. ADAPTING REGULARIZATION

Our aim is to adapt $\boldsymbol{\kappa}$ so as to minimize the validation error. We can apply the iterative gradient descent scheme originally suggested in [10]:

$$\boldsymbol{\kappa}^{(j+1)} = \boldsymbol{\kappa}^{(j)} - \eta \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{(j)})) \tag{3}$$

where η is a line search parameter and $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{(j)})$ is the estimated weight vector using $\boldsymbol{\kappa}^{(j)}$. The regularization term $R(\boldsymbol{w}, \boldsymbol{\kappa})$ is supposed to be linear in $\boldsymbol{\kappa}$:

$$R(\boldsymbol{w}, \boldsymbol{\kappa}) = \boldsymbol{\kappa}^\top \boldsymbol{r}(\boldsymbol{w}) = \sum_{i=1}^{q} \kappa_i r_i(\boldsymbol{w}) \tag{4}$$

where κ_i are the regularization parameters and $r_i(\boldsymbol{w})$ the associated regularization functions. In these conditions, the gradient of the validation error becomes [10], [11]:

$$\frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}) = -\frac{\partial \boldsymbol{r}}{\partial \boldsymbol{w}^\top}(\widehat{\boldsymbol{w}}) \cdot \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{w}}(\widehat{\boldsymbol{w}}), \tag{5}$$

where $\boldsymbol{J} = \partial^2 C / \partial \boldsymbol{w} \partial \boldsymbol{w}^\top$ is the Hessian matrix of the cost function. Suppose that the weight vector is partitioned into q groups $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_q)$ and we use one weight decay parameter κ_i for each group, i.e., $R(\boldsymbol{w}, \boldsymbol{\kappa}) = \sum_{i=1}^{q} \kappa_i |\boldsymbol{w}_i|^2$. In this case, the gradient yields:

$$\frac{\partial S_{\mathcal{V}}}{\partial \kappa_i}(\widehat{\boldsymbol{w}}) = -2(\widehat{\boldsymbol{w}}_i)^\top \cdot \boldsymbol{s}_i \tag{6}$$

where $\boldsymbol{s} = [\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_q] = \boldsymbol{J}^{-1}(\widehat{\boldsymbol{w}}) \cdot \partial S_{\mathcal{V}}(\widehat{\boldsymbol{w}}) / \partial \boldsymbol{w}$. In order to ensure that $\kappa_i \geq 0$ we perform a re-parameterization,

$$\kappa_i = \begin{cases} \exp(\lambda_i) & , \lambda_i < 0 \\ \lambda_i + 1 & , \lambda_i \geq 0 \end{cases} \tag{7}$$

and carry out the minimization w.r.t. the new parameters $\boldsymbol{\lambda}$. Note that $\partial S_{\mathcal{V}} / \partial \lambda_i = \partial \kappa_i / \partial \lambda_i \cdot \partial S_{\mathcal{V}} / \partial \kappa_i$.

In order to improve convergence we suggest to use the Polak-Ribiere conjugate method. Let $\boldsymbol{g}^{(j)}$ be the gradient at the current iteration j:

$$\boldsymbol{g}^{(j)} = \frac{\partial S_{\mathcal{V}}}{\partial \boldsymbol{\kappa}}(\widehat{\boldsymbol{w}}(\boldsymbol{\kappa}^{(j)})) \tag{8}$$

The search direction $\boldsymbol{h}^{(j)}$ is updated as follow:

$$\begin{aligned} \boldsymbol{h}^{(j)} &= -\boldsymbol{g}^{(j)} + \gamma_{j-1} \cdot \boldsymbol{h}^{(j-1)} && (9) \\ \gamma_{j-1} &= \frac{(\boldsymbol{g}^{(j)})^\top \cdot (\boldsymbol{g}^{(j)} - \boldsymbol{g}^{(j-1)})}{(\boldsymbol{g}^{(j-1)})^\top \cdot \boldsymbol{g}^{(j-1)}} && (10) \end{aligned}$$

Once the search direction $\boldsymbol{h}^{(j)}$ has been calculated, a line search is performed in order to find a set of parameters that lead to a significant decrease in the cost function. The traditional method involves a bracketing of the minimum followed by a combination of golden section search and parabolic interpolation to close in on the minimum. In such a scheme, most function evaluations are performed during the line search. We prefer to implement an approximate line search combined with the Wolfe-Powell stop condition [14, App. B]. Prospective parameters are obtained by a combination of section search and third order polynomial interpolation and extrapolation. The line search stops when the current function value is significantly smaller than what we started with, while the slope is only a fraction of the initial slope.

It has been argued [2], [13] that the line search could be performed efficiently without derivatives. While there are some arguments in favor of this claim, we favor a line search with derivatives, for two main reasons: 1) the stop condition for the approximate line search involves the slope, hence the derivatives, and 2) the gradient will be needed to calculate the next search direction.

In the comparison of section 4, the steepest descent algorithm uses the same line search.

In summary, the adaptive regularization algorithm is:

1. Select the split ratio γ and initialize $\boldsymbol{\kappa}$, and the weights of the network.
2. Train the network with fixed $\boldsymbol{\kappa}$ to achieve $\widehat{\boldsymbol{w}}(\boldsymbol{\kappa})$. Calculate the validation error $S_{\mathcal{V}}$.
3. Calculate the gradient $\partial S_{\mathcal{V}} / \partial \boldsymbol{\kappa}$ using Eq. (5).

4. Calculate the search direction using Eq. (9).
5. Perform an approximate line search in the direction $\boldsymbol{h}^{(j)}$ to find a new $\boldsymbol{\kappa}$.
6. Repeat steps 2–5 until either the relative change in validation error is below a small percentage or the gradient is close to 0.

4. EXPERIMENTS

We test the performance of the conjugate gradient algorithm for adapting regularization parameters on artificial data generated by the system described in [4, Sec. 4.3]:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - \tfrac{1}{2})^2 + 10x_4 + 5x_5 + \varepsilon \quad (11)$$

where the inputs are uniformly distributed $x_i \sim \mathcal{U}(0,1)$ and the noise is Gaussian distributed $\varepsilon \sim \mathcal{N}(0,1)$. The data set consisted of $N = 200$ examples with 10 dimensional input vector $\boldsymbol{x}$. Inputs $x_6, \cdots, x_{10}$ are $\mathcal{U}(0,1)$ and do not convey relevant information for the output y, cf. Eq. (11). The data set were split into $N_t = 100$ for training and $N_v = 100$ for validation. In addition, we generated a test set of $N_{\text{test}} = 4000$ samples.

In our simulations, we used a feed-forward neural network model with 10 inputs and 5 hidden units with hyperbolic tangent activations. Training is done by minimizing the quadratic loss function, augmented with weight decay regularizers. All weights from one input have an associated weight decay parameter $\kappa_1, \cdots, \kappa_{10}$, and the hidden-to-output weights have a weight-decay parameter κ_{11}.

Weights were initialized uniformly over the interval $[-0.5/\sqrt{f}, 0.5/\sqrt{f}]$, where f is the "fan-in", i.e., the number of incoming weights to a given unit. Regularization parameters are first initialized to 10^{-6}. The network is then trained for 10 iterations, after which the κ_i are set to $\nu_{\max}/10^4$, where $\nu_{\max}$ is the maximum eigenvalue of the Hessian matrix of the cost function. This prevents numerical stability problems.

Weights are estimated using the conjugate gradient algorithm and the regularization parameters are adapted using the algorithm in Sec. 3. The inverse Hessian required in Eq. (5) is found as the Moore-Penrose pseudo inverse (see e.g., [15]) ensuring that the eigenvalue spread is less than 10^8, i.e., the square root of the machine precision [3]. $\boldsymbol{J}$ is estimated using the Gauss-Newton approximation [15].

Weights are finally retrained on the combined set of training and validation data using the optimized weight decay parameters.

Table 1 reports the average and standard deviations of the errors over 5 runs for different initializations.

<table>
<tr><th></th><th>Neural Network</th><th>Flexible Kernel</th><th>Linear Model</th></tr>
<tr><td>Train.</td><td>0.92 ± 0.11</td><td rowspan="2">1.22</td><td rowspan="2">5.06</td></tr>
<tr><td>Val.</td><td>1.79 ± 0.13</td></tr>
<tr><td>Test</td><td>3.01 ± 0.30</td><td rowspan="2">5.96</td><td rowspan="2">7.93</td></tr>
<tr><td>Test after retrain.</td><td>2.26 ± 0.18</td></tr>
</table>

Table 1: Training, validation and test errors. For the neural network the averages and standard deviations are over 5 runs. For comparison we listed the performance of a linear model and of a kernel smoother with a diagonal smoothing matrix [16] optimised by minimizing the leave-one-out cross-validation error.

Note that retraining on the combined data set decreases the test error somewhat on the average.

Fig. 1 shows a typical run of the $\boldsymbol{\kappa}$ adaptation algorithm as well as a comparison with a simple steepest descent method.

5. DISCUSSION

Our experience with adaptive regularization is globally very positive. Combined with an efficient multidimensional minimization method like the conjugate gradient algorithm, it allows for a reliable adaptation of the regularization parameter.

Furthermore, it is flexible enough to allow a wide class of regularization. We have here shown how this scheme can be used to estimate the relevance of the input. This is similar in spirit to the *Automatic Relevance Determination* of Neal and MacKay [12].

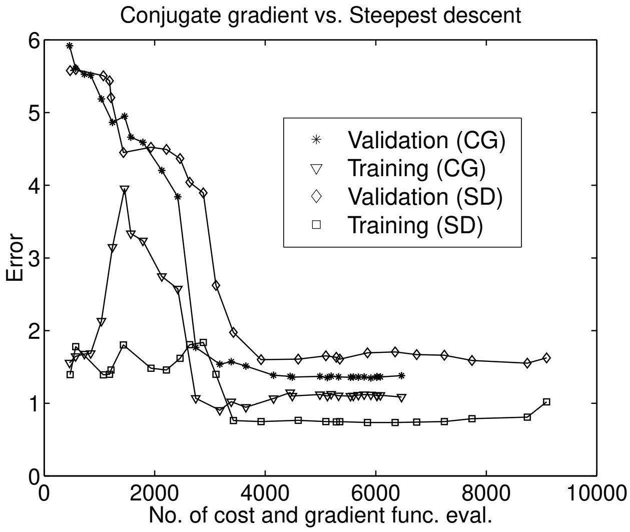
6. CONCLUSIONS

This paper presented an improved algorithm for adaptation of regularization parameters. Numerical examples demonstrated the potential of the framework.

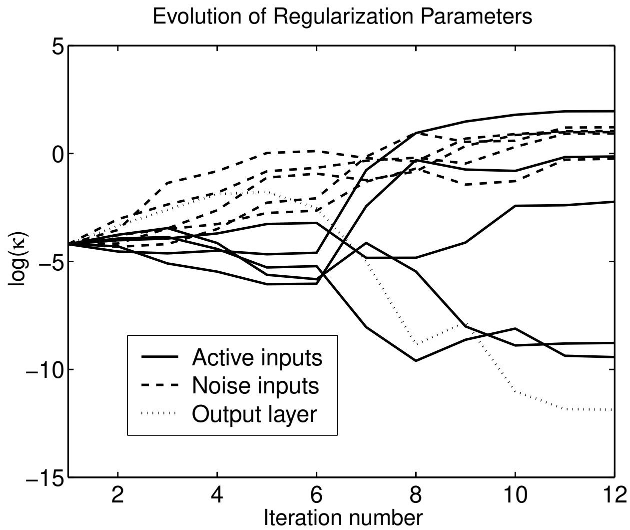
7. REFERENCES

[1] L.N. Andersen, J. Larsen, L.K. Hansen & M Hintz-Madsen: "Adaptive Regularization of Neural Classifiers," in J. Principe *et al.* (eds.) *Proc. IEEE Workshop on Neural Networks for Signal Processing VII*, Piscataway, New Jersey: IEEE, pp. 24–33, 1997.

[2] C.M. Bishop: *Neural Networks for Pattern Recognition*, Oxford, UK: Oxford University Press, 1995.



(a)



(b)

Figure 1: Typical run of the $\boldsymbol{\kappa}$ adaptation algorithm using either steepest descent (SD) or conjugate gradient (CG). Panel (a): training and validation errors in both cases. Note that CG both converges faster and yield slightly lower validation error. The total number of cost and gradient evaluation is a good measure of the total computational burden. Panel (b): evolution of the log-weight decay parameters using conjugate gradient. Most active inputs have small weight decays, while the noise inputs have higher weight decays. However, notice that the overall influence is determined by the weight decay as well as the value of the weights. The output layer weight decay is seemingly not important.

[3] J.E. Dennis & R.B. Schnabel: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, New Jersey: Prentice-Hall, 1983.

[4] J.H. Friedman: "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[5] S. Geman, E. Bienenstock & R. Doursat: "Neural Networks and the Bias/Variance Dilemma," *Neural Computation*, vol. 4, pp. 1–58, 1992.

[6] K. Hornik: "Approximation Capabilities of Multilayer Feedforward Networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[7] P.J. Huber: *Robust Statistics*, New York, New York: John Wiley & Sons, 1981.

[8] M. Kearns: "A Bound on the Error of Cross Validation Using the Approximation and Estimation Rates, with Consequences for the Training-Test Split," *Neural Computation*, vol. 9, no. 5, pp. 1143–1161, 1997.

[9] J. Larsen & L.K. Hansen: "Empirical Generalization Assessment of Neural Network Models," in F. Girosi *et al.* (eds.), *Proc. IEEE Workshop on Neural Networks for Signal Processing V*, Piscataway, New Jersey: IEEE, 1995, pp. 30–39.

[10] J. Larsen, L.K. Hansen, C. Svarer & M. Ohlsson: "Design and Regularization of Neural Networks: The Optimal Use of a Validation Set," in S. Usui *et al.* (eds.), *Proc. IEEE Workshop on Neural Networks for Signal Processing VI*, Piscataway, New Jersey: IEEE, 1996, pp. 62–71.

[11] J. Larsen, C. Svarer, L.N. Andersen & L.K. Hansen: "Adaptive Regularization in Neural Network Modeling," appears in G.B. Orr *et al.* (eds.) *"The Book of Tricks"*, Germany: Springer-Verlag, 1997. Available by `ftp://eivind.mm.dtu.dk/dist/1997/larsen.bot.ps.Z`.

[12] R.M. Neal: *Bayesian Learning for Neural Networks*, New York: Springer Verlag, 1996.

[13] W.H. Press, S.A. Teukolsky, W.T. Vetterling, & B.P. Flannery: *Numerical Recipes in C, The Art of Scientific Computing*, Cambridge, Massachusetts: Cambridge University Press, 2nd Edition, 1992.

[14] Carl E. Rasmussen: *Evaluation of Gaussian Processes and Other Methods for Non-Linear Regression*, Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, 1996. Available by: `ftp://ftp.cs.toronto.edu/pub/carl/thesis.ps.gz`.

[15] G.A.F. Seber & C.J. Wild: *Nonlinear Regression*, New York, New York: John Wiley & Sons, 1989.

[16] M.P. Wand & M.C. Jones: *Kernel Smoothing*, New York, New York: Chapman & Hall, 1995.