A GENTLE HESSIAN FOR EFFICIENT GRADIENT DESCENT

Ronan Collobert and Samy Bengio

IDIAP, Martigny, Switzerland
{collober,bengio}@idiap.ch

ABSTRACT

Several second-order optimization methods for gradient descent algorithms have been proposed over the years, but they usually need to compute the inverse of the Hessian of the cost function (or an approximation of this inverse) during training. In most cases, this leads to an $O(n^2)$ cost in time and space per iteration, where n is the number of parameters, which is prohibitive for large n. We propose instead a study of the Hessian *before* training. Based on a second order analysis, we show that a block-diagonal Hessian yields an easier optimization problem than a full Hessian. We also show that the condition of block-diagonality in common machine learning models can be achieved by simply selecting an appropriate training criterion. Finally, we propose a version of the SVM criterion applied to MLPs, which verifies the aspects highlighted in this second order analysis, but also yields very good generalization performance in practice, taking advantage of the margin effect. Several empirical comparisons on two benchmark datasets are given to illustrate this approach.

1. INTRODUCTION

Optimization by gradient descent is widely used by various machine learning algorithms such as back-propagation of the error in Multi-Layer Perceptrons (MLPs) and Radial Basis Functions [1]. Unfortunately, empirical evidences show that results obtained after training a model by gradient descent are often highly variable. Hence, in the last few decades, several researchers proposed various enhancements [2] to classical gradient descent algorithms. Most of these enhancements focus on variations of second-order optimization methods [3], and thus have to compute at each training iteration the inverse of the Hessian[1] of the cost function. Therefore, the time complexity of the resulting algorithms grows in $O(n^3)$ per iteration, and in $O(n^2)$ in space, where n is the number of parameters. Thus these algorithms become useless for very large datasets and models. Some enhancements which compute iteratively the inverse of the Hessian have been proposed but most of them still have a cost in $O(n^2)$ per iteration. In the end, most of the time, people rely on simple stochastic gradient descent which has a cost in $O(n)$ per iteration, and which in general outperforms most other methods on large problems [2].

Instead of dealing with the Hessian during training as several other methods do, we propose a study of the Hessian of the cost function *before* training. Thus, after a presentation of the framework in Section 2, we analyze in Section 3 the cost function, using a second order Taylor approximation. We show that a block-diagonal Hessian yields an easier optimization problem than a full Hessian. We also illustrate this with the case of MLPs where the condition of block-diagonality can be achieved by simply selecting an appropriate training criterion. Then, in Section 4, we propose a version of the Support Vector Machine (SVM) criterion applied to MLPs, which verifies the aspects highlighted in this second order analysis, but also yields very good generalization performance in practice, taking advantage of the margin effect.

The authors acknowledge financial support from the Swiss NCCR (IM)2 project.

[1]which is the second order derivative of the cost with respect to pairs of parameters.

2. FRAMEWORK

We consider two-class classification problems: given a training set of T examples $(\mathbf{x}_t, y_t)_{t=1\dots T}$ with $(\mathbf{x}_t, y_t) \in \mathbb{R}^d \times \{-1, 1\}$ where $\mathbf{x}_t$ is the input vector of the t^{th} example, and y_t is the corresponding class, we would like to find a function $f(\cdot)$ such that

$$\left.\begin{array}{ll} f(\mathbf{x}_t) > 0 & \text{when } y_t = 1 \\ f(\mathbf{x}_t) < 0 & \text{when } y_t = -1 \end{array}\right\} \quad \forall t \; . \tag{1}$$

Our interest is to study functions which can be trained by gradient descent techniques. Due to the lack of space, we will focus only on Multi-Layer Perceptrons (MLPs), but this work can be extended to other models [4]. The MLP we consider here has one hidden layer of N units:

$$f(\mathbf{x}) = b + \sum_{n=1}^{N} \alpha_n \, h(\mathbf{w}_n^T \, \mathbf{x}) \tag{2}$$

where $\mathbf{w}_n \in \mathbb{R}^d$ are the weights of the hidden layer,[2] $\alpha_n \in \mathbb{R}$ are the weights of the output layer, and b is the bias of the output layer. $h(\cdot)$ is a transfer function which is usually a hyperbolic tangent. It has been shown in [5] that MLPs with one hidden layer and hyperbolic tangent transfer functions are universal approximators of real valued functions. This means that there exist at least one MLP such as in (2) which satisfies conditions (1).

2.1. Training with Gradient Descent

Given a model $f_{\boldsymbol{\theta}}(\cdot)$ which could be an MLP, we select a cost function $C(f_{\boldsymbol{\theta}}(\mathbf{x}), \, y)$, and we minimize the cost

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t=1}^{T} C(f_{\boldsymbol{\theta}}(\mathbf{x}_t), \, y_t) \; , \tag{3}$$

[2]To simplify the notation, we suppose here that the last coordinate of the $\mathbf{x}$ vector is 1, and thus the bias of unit n is represented by the last coordinate of $\mathbf{w}_n$.

using stochastic gradient descent. One of the most common cost functions used for classification is the mean-squared error (MSE):

$$C(f_{\boldsymbol{\theta}}(\mathbf{x}),\, y) = \frac{1}{2}(y - f_{\boldsymbol{\theta}}(x))^2 \ .$$

It can be shown [1] that with an infinite amount of data, the minimum of the MSE criterion is obtained when $f_{\boldsymbol{\theta}}(\mathbf{x})$ is equal to the true posterior probability[3] $p(y|\mathbf{x})$. It has also been shown [6] that taking the decision which maximizes $p(y|\mathbf{x})$ leads to the minimum classification error rate. Hence, the use of the MSE criterion is relevant for classification. However, in a likelihood framework, minimizing the MSE criterion is equivalent to maximizing a likelihood under the hypothesis that y is generated from a smooth function with added Gaussian noise. Since y is a binary variable, some researchers prefer to consider y as coming from a Bernoulli distribution, which leads to another well-known criterion, often called "cross-entropy" (CE) [1]. This one can be rewritten as the following, in the case of a two-class classification problem:

$$\begin{aligned} C(f_{\boldsymbol{\theta}}(\mathbf{x}),\, y) &= -\log p_{\boldsymbol{\theta}}(y|\mathbf{x}) \\ &= \log(1 + \exp(-y f_{\boldsymbol{\theta}}(\mathbf{x}))) \ . \end{aligned} \tag{4}$$

Once again, with an infinite amount of data, $p_{\boldsymbol{\theta}}(y|\mathbf{x})$ which maximizes the likelihood tends to the true posterior probability $p(y|\mathbf{x})$. Thus, as for the MSE criterion, the CE criterion minimizes ultimately the classification error.

2.2. Experimental Setup

Experiments shown in this paper have been performed using the two biggest datasets available on the *UCI* web site. The first one is *UCI Forest*. We modified the 7-class classification problem into a balanced binary classification problem where the goal was to separate class 2 from the others. We used 100,000 examples for training, 10,000 for validation and 50,000 for testing. The second dataset is *UCI Connect-4*. We modified the 3-class classification problem into a balanced binary classification problem where the goal was to separate class "won" against the others. We used 50,000 examples for training, 7,000 for validation and 10,000 for testing. Validation sets were only used to select hyper-parameters of the models in Section 4.

3. LOCAL BEHAVIOR OF COST FUNCTIONS

In this section, we will focus only on the *training* performance of several architectures, trained with gradient descent. Table 1 shows

Criterion	Train Err. (%)		Train MSE	
	Forest	Connect-4	Forest	Connect-4
MSE	13.0	8.2	0.41	0.31
CE	10.3	0.0	0.30	0.01

Table 1. Train errors for MLPs trained with different criterions.

a preliminary comparison of an MLP trained using the MSE and CE criteria. The classification error rate has been chosen to compare performances because both MSE and CE criteria tend to minimize the classification error, as highlighted in the previous section.

[3]with class labels $y \in \{0, 1\}$, which can be obtained in our case by simple rescaling.

We added an evaluation of the MSE error[4] to be more convincing. We chose an arbitrary large number (500) of hidden units. Moreover, all other hyper-parameters, such as the learning rate, were also selected according to the training set. For both datasets, the training performance of an MLP trained with MSE criterion is statistically significantly worse (with 99% confidence) than an MLP trained with the CE criterion. The fact that, on these two datasets, an MLP trained with the CE criterion has a significantly lower MSE error than an MLP trained with the MSE criterion clearly denotes *an optimization problem with the MLP trained with the MSE criterion*.

3.1. Second Order Optimization Algorithms

To understand these differences of performances, we propose to study the *local* behavior of each model and its corresponding cost function. Given a model $f_{\boldsymbol{\theta}}(\cdot)$ where we want to optimize parameters $\boldsymbol{\theta}$, and given a vector of parameters $\boldsymbol{\theta}^o$, the cost function $E_{\mathbf{x},y}(\boldsymbol{\theta}) = C(f_{\boldsymbol{\theta}}(\mathbf{x}),\, y))$ can be approximated with respect to $\boldsymbol{\theta}$ around $\boldsymbol{\theta}^o$, by a second order Taylor expansion:

$$\begin{aligned} E_{\mathbf{x},y}(\boldsymbol{\theta}) &= E_{\mathbf{x},y}(\boldsymbol{\theta}^o) \\ &\quad + (\boldsymbol{\theta} - \boldsymbol{\theta}^o)^T \, \frac{\partial E_{\mathbf{x},y}(\boldsymbol{\theta}^o)}{\partial \boldsymbol{\theta}} \\ &\quad + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^o)^T \, H_{\mathbf{x},y}(\boldsymbol{\theta}^o) \, (\boldsymbol{\theta} - \boldsymbol{\theta}^o) \\ &\quad + o(\|\boldsymbol{\theta} - \boldsymbol{\theta}^o\|_2^2) \end{aligned} \tag{5}$$

where $\partial E_{\mathbf{x},y}(\boldsymbol{\theta}^o)/\partial \boldsymbol{\theta}$ and $H_{\mathbf{x},y}(\boldsymbol{\theta}^o)$ are respectively the gradient and the Hessian matrix of $E_{\mathbf{x},y}$ with respect to $\boldsymbol{\theta}$, evaluated at $\boldsymbol{\theta}^o$. We use $\|.\|_2$ as the Euclidean norm for vectors, and $o(\|\boldsymbol{\theta} - \boldsymbol{\theta}^o\|_2^2)$ to represent a term negligible with respect to $\|\boldsymbol{\theta} - \boldsymbol{\theta}^o\|_2^2$.

Because of the lack of space we will not focus on the first derivative in this paper. Our main concern will be the study of the Hessian.

3.2. Advantage of a Block Diagonal Hessian

Let us consider a model $f_{\boldsymbol{\theta}}(\cdot)$ where the parameter vector $\boldsymbol{\theta}$ can be segmented into several sub-vectors $\boldsymbol{\theta} = (\boldsymbol{\theta}_1,\, \boldsymbol{\theta}_2 \ldots \boldsymbol{\theta}_n)$. Then, given a current state $\boldsymbol{\theta}^o$, and if we we forget negligible terms with respect to $\|\boldsymbol{\theta} - \boldsymbol{\theta}^o\|_2^2$, the local quadratic approximation (5) can be rewritten as:

$$\begin{aligned} E_{\mathbf{x},y}(\boldsymbol{\theta}) &= E_{\mathbf{x},y}(\boldsymbol{\theta}^o) \\ &\quad + \sum_i (\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^o)^T \, \frac{\partial E_{\mathbf{x},y}(\boldsymbol{\theta}^o)}{\partial \boldsymbol{\theta}} \\ &\quad + \sum_{i,j} \frac{1}{2}(\boldsymbol{\theta}_i - \boldsymbol{\theta}_i^o)^T \, H^{i,j}_{\mathbf{x},y}(\boldsymbol{\theta}^o) \, (\boldsymbol{\theta}_j - \boldsymbol{\theta}_j^o) \ , \end{aligned} \tag{6}$$

where $H^{i,j}_{\mathbf{x},y}$ is the matrix $\frac{\partial^2 E_{\mathbf{x},y}}{\partial \boldsymbol{\theta}_i \, \partial \boldsymbol{\theta}_j}$ using vectorial derivation. In the ideal case where $H_{\mathbf{x},y}$ is block-diagonal, that is if $H^{i,j}_{\mathbf{x},y} = 0$ for $i \neq j$, this leads to

$$E_{\mathbf{x},y}(\boldsymbol{\theta}) = E_{\mathbf{x},y}(\boldsymbol{\theta}^o) + \sum_i E^i_{\mathbf{x},y}(\boldsymbol{\theta}_i) \tag{7}$$

[4]In order to have comparable results, we rescaled the output probability $p_{\boldsymbol{\theta}}(y|\mathbf{x})$ of the MLP trained with the CE criterion between -1 and 1.

where

$$E^i_{\mathbf{x},y}(\boldsymbol{\theta}_i) = (\boldsymbol{\theta}_i - \boldsymbol{\theta}^o_i)^T \frac{\partial E_{\mathbf{x},y}(\boldsymbol{\theta}^o)}{\partial \boldsymbol{\theta}} + \frac{1}{2}(\boldsymbol{\theta}_i - \boldsymbol{\theta}^o_i)^T H^{i,i}_{\mathbf{x},y}(\boldsymbol{\theta}^o)\,(\boldsymbol{\theta}_i - \boldsymbol{\theta}^o_i)\ .$$

Equation (7) shows that the error function $E_{\mathbf{x},y}(\boldsymbol{\theta})$ can be split into n independent error functions $E^i_{\mathbf{x},y}(\boldsymbol{\theta}_i)$. In other words, the optimization of a sub-vector $\boldsymbol{\theta}_i$ is locally independent of the others $\boldsymbol{\theta}_j,\ j \neq i$. Therefore, the optimization problem is *much simpler* than with a full Hessian where the modification of only one parameter would also affect the modification of all others.

If $H_{\mathbf{x},y}$ is not truly block-diagonal, it can be shown that the more the spectral norm $\|H^{i,j}_{\mathbf{x},y}\|_2$ of the blocks of the Hessian outside the diagonal tends to zero, the more equation (7) is accurate, and the more training of each sub-vectors $\boldsymbol{\theta}_i$ becomes independent. Generally speaking, we can conclude that the more the Hessian is block-diagonal, the easier should be the training of the model.

3.3. Illustration

Let us now analyze the Hessian of the MLP given in (2) that would be respectively trained with the MSE and CE criteria. First the Hessian when trained with the MSE criterion:

$$\frac{\partial^2 E_{\mathbf{x},y}}{\partial \mathbf{w}_i\,\partial \mathbf{w}_j} = \alpha_i \alpha_j\, h'(\mathbf{w}_i^{\mathbf{T}}\,\mathbf{x})\, h'(\mathbf{w}_j^{\mathbf{T}}\,\mathbf{x})\, \mathbf{x}\mathbf{x}^{\mathbf{T}} \quad (i \neq j)\ .$$

Note that there is no obvious reason for this Hessian to tend to zero, whereas if we compute the Hessian with the cross-entropy (CE) criterion we get:

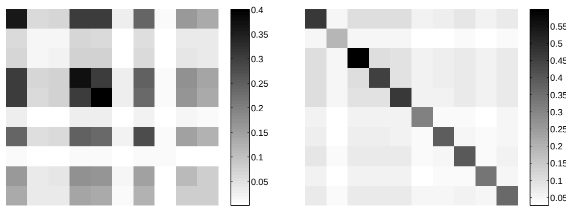
$$\frac{\partial^2 E_{\mathbf{x},y}}{\partial \mathbf{w}_i\,\partial \mathbf{w}_j} = p_{\boldsymbol{\theta}}(y|\mathbf{x}) p_{\boldsymbol{\theta}}(-y|\mathbf{x}) \times \alpha_i \alpha_j\, h'(\mathbf{w}_i^{\mathbf{T}}\,\mathbf{x}) h'(\mathbf{w}_j^{\mathbf{T}}\,\mathbf{x})\, \mathbf{x}\mathbf{x}^{\mathbf{T}} \quad (i \neq j)\ .$$

Here the term $p_{\boldsymbol{\theta}}(y|\mathbf{x})p_{\boldsymbol{\theta}}(-y|\mathbf{x}) = p_{\boldsymbol{\theta}}(y|\mathbf{x})(1 - p_{\boldsymbol{\theta}}(y|\mathbf{x}))$ will tend very quickly to zero, since we are training the MLP to maximize $p_{\boldsymbol{\theta}}(y|\mathbf{x})$. It will push the Hessian obtained with the CE criterion toward almost block-diagonality (one block for each pair of units, see Figure 1(b)), whereas the Hessian obtained with the MSE criterion remains full, as shown in Figure 1(a).

Note also that some researchers proposed to add a hyperbolic tangent at the output layer of the MLP proposed in (2) to improve classification performances, when training with an MSE criterion. With similar derivations, it is possible to show that this hyperbolic tangent tends to improve the block-diagonality of the Hessian, but with the drawback that this block-diagonality corresponds to a zero first derivative of the cost function with respect to the weights. Thus, we observed that the performances were in practice significantly worse than the performances of an MLP trained with the cross-entropy criterion (see [4] for more details).

4. SVM MARGIN FOR MLPS

Support Vector Machines (SVMs) [7] generally yield good performance as compared to other algorithms. Given the two-class classification problem presented in Section 2, a linear SVM finds a separating hyper-plane which maximizes the margin between this hyper-plane and the two classes. Thus, the SVM solution is a trade-off between maximization of the margin and minimization of the classification error. More formally, given a hyper-plane $f_{\boldsymbol{\alpha},b}(\mathbf{x}) = 0$ with $f_{\boldsymbol{\alpha},b}(\mathbf{x}) = \boldsymbol{\alpha}\,\mathbf{x} + b$ ($\boldsymbol{\alpha} \in \mathbb{R}^d,\ b \in \mathbb{R}$), the SVM problem is equivalent to minimize



(a) Hessian for MSE (b) Hessian for CE

Fig. 1. Description of the Hessian matrix obtained with an MLP trained with the MSE criterion in (a) and with the CE criterion in (b). The Hessian has been averaged over all the examples. MLPs have 10 hidden units. Each block corresponds to a pair of hidden units, and is represented by its spectral norm. Results on the Forest dataset, with 10,000 training examples, after 5 iterations.

$$L(\boldsymbol{\alpha},\, b) = \frac{\mu}{2}\, \|\boldsymbol{\alpha}\|_2^2 + \sum_{t=1}^{T} |1 - y_t f_{\boldsymbol{\alpha},b}(\mathbf{x}_t)|_+ \qquad (8)$$

where $|z|_+ = \max(0, z)$, and $\mu \in R^+$ is a constant which acts as a trade-off between the first term which corresponds to the margin maximization, and the second term which tries to force the two classes to be separated. Non-linear SVMs are obtained by projecting input vectors in a higher dimensional space, and by maximizing the margin *in this higher dimensional space*, using the so-called kernel trick. As MLP proposed in (2) first sends input vectors into a non-linear space using the hidden layer, and then separates the data in this space using the output layer, we could maximize the margin in the non-linear space of the MLPs, using the stochastic version of the criterion (8), as already suggested in [8]. Unfortunately, even if this criterion led in practice to similar training performances as compared to the CE criterion for $\mu = 0$, the training was significantly slowed down for $\mu > 0$. This could be explained with the fact that when $\mu > 0$, we force the output weights of the MLP to be small, which reduces the gradient received by the hidden units. In order to fix this problem, we propose to use instead the following cost function:

$$C(f(\mathbf{x}),\, y) = |\beta - y f(\mathbf{x})|_+ \qquad (9)$$

where $\beta \in \mathbb{R}^+$ is a hyper-parameter similar to μ, which controls the trade-off between the margin maximization and the separation conditions. After some arithmetics, we obtain the margin $\frac{2\beta}{\|\boldsymbol{\alpha}\|_2}$. In order to guarantee that we increase this margin by increasing β, we can fix the norm $\|\boldsymbol{\alpha}\|_2$ to an arbitrary chosen value, but there is an even simpler solution: we can fix the output weights $\boldsymbol{\alpha}$ to the same value (no training). This has a sense in our case, because it has been shown that such an MLP can approximate any kind of boolean functions, and thus, can be applied to classification [9]. In practice, these two techniques to increase the margin gave similar results in training and generalization performances. The results that will be shown in the following sections were produced by fixing the output weights to a constant.

4.1. Training Performances

Not considering the rare case $yf(\mathbf{x}) = 1$ where the cost function is not differentiable, we can derive the following Hessian:

$$\frac{\partial^2 E_{\mathbf{x},y}}{\partial \mathbf{w}_i \, \partial \mathbf{w}_j} = 0 \quad (i \neq j) \ .$$

The Hessian of the cost function is thus *completely block-diagonal*, which guarantees the local independence in the training of the hidden units. This is verified in practice on the training performances, which are similar to the training performances obtained with a CE criterion, when we do not maximize the margin, that is, for small values of β.

4.2. Improving Generalization Performances

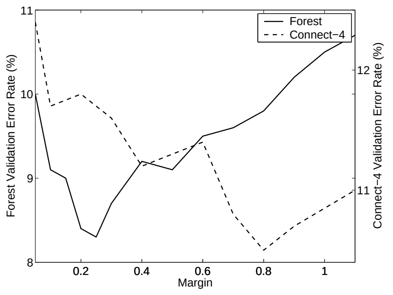


Fig. 2. Evolution of the validation error with respect to the margin for an MLP with outputs weights $\boldsymbol{\alpha}$ fixed to 1. Results on Forest dataset.

Following the theory of Section 4, there is in practice an optimal size of the margin as suggested in Figure 2. The best generalization performances (shown in Table 2), obtained after tuning the margin and all other hyper-parameters according to the validation set, were 8.5% error on the test set of Forest and 10.3% error on the test set of Connect-4. These are great improvements (statistically significant with 99% confidence) compared to a standard CE criterion which obtained respectively 11.1% and 11.4% of testing error rates. Moreover, well tuned SVMs with RBF kernel trained using the fast SVMTorch package are significantly worse, larger and slower on these two tasks.

5. CONCLUSION

In this paper we have analyzed gradient descent algorithms with respect to several important aspects. As already known, the Hessian plays a major role in the effectiveness of any gradient descent algorithm. We explained why a block-diagonal Hessian should yield more efficient training algorithms. We showed on common models how the choice of the training criterion influences the Hessian matrix, and hence how to select an efficient training criterion. Finally, we introduced a new cost function for MLPs, inspired by the SVM algorithm which yields a block-diagonal Hessian and enables the control of the margin in the hidden layer space. This cost function yielded (statistically significantly) better generalization performance on two benchmark datasets in much less time than SVMs. This work shows that a carefully tuned gradient descent can still be competitive and even outperform recent machine learning algorithms in training and generalization performance, but also in training time.

6. REFERENCES

[1] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

[2] Y. LeCun, L. Bottou, G.B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*, G.B. Orr and K.-R. Müller, Eds., pp. 9–50. Springer, 1998.

[3] T. Battiti, "First and second-order methods for learning: Between steepest descent and Newton's method," *Neural Computation*, vol. 4, no. 2, pp. 141–166, 1992.

[4] R. Collobert and S. Bengio, "A new margin-based criterion for efficient gradient descent," Technical Report IDIAP-RR 03-16, IDIAP, 2003.

[5] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.

[6] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[7] V.N. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[8] S. Zhong and J. Ghosh, "Decision boundary focused neural network classifier," in *Intelligent Engineering Systems Through Artificial Neural Networks (ANNIE)*. 2000, ASME.

[9] P. Auer, H. Burgsteiner, and W. Maass, "Reducing communication for distributed learning in neural networks," in *ICANN'2002*, J. R. Dorronsoro, Ed. 2002, vol. 2415 of *Lecture Notes in Computer Science*, pp. 123–128, Springer.

Model	Cost Function	Test Err. (%)		HU		Time Factor	
		Forest	Connect-4	Forest	Connect-4	Forest	Connect-4
SVM	SVM	12.2	12.6	34291	18156	2.7	3.6
MLP	MSE	13.9	12.8	200	500	0.4	1.0
MLP	CE	11.1	11.4	500	500	1.0	1.0
MLP	**Margin**	**8.5**	**10.3**	500	500	1.0	1.0

Table 2. Test errors of several models, including SVMs. Note that all hyper-parameters, especially the number of hidden units were chosen according to validation sets.