Ho-Kashyap Classifier with Early Stopping for Regularization

Fabien Lauer, Gérard Bloch

HAL Id: hal-00021551

https://hal.archives-ouvertes.fr/hal-00021551

Submitted on 22 Mar 2006

Ho–Kashyap Classifier with Early Stopping for Regularization

Fabien Lauer and Gérard Bloch [*]

Université Henri Poincaré – Nancy 1 (UHP), Centre de Recherche en Automatique de Nancy (CRAN UMR CNRS 7039), CRAN–ESSTIN, Rue Jean Lamour, 54519 Vandœuvre Cedex, France

Abstract

This paper focuses on linear classification using a fast and simple algorithm known as the Ho–Kashyap learning rule (HK). In order to avoid overfitting and instead of adding a regularization parameter in the criterion, early stopping is introduced as a regularization method for HK learning, which becomes HKES (Ho–Kashyap with Early Stopping). Furthermore, an automatic procedure, based on generalization error estimation, is proposed to tune the stopping time. The method is then tested and compared to others (including SVM and LSVM), that use either ℓ_1 or ℓ_2-norm of the errors, on well-known benchmarks. The results show the limits of the early stopping for regularization with respect to the generalization error estimation and the drawbacks of low level hyperparameters such as a number of iterations.

Key words: Classifier design, Generalization, Ho–Kashyap classifier, Early stopping, Robustness

1 Introduction

In a linear classification problem, one often tend to use Support Vector Machine (SVM) classifiers [1–3] which appear to be the state-of-the-art in matter of classification. But such tools often result in considerably high calculation times though many efforts have been done and new algorithms proposed to solve that problem [4–6].

---

[*] Corresponding author.
Email addresses: `fabien.lauer@esstin.uhp-nancy.fr` (Fabien Lauer), `gerard.bloch@esstin.uhp-nancy.fr` (Gérard Bloch).

This paper focuses on linear classification using a rather simple and fast gradient descent based algorithm known as the Ho–Kashyap learning rule (HK) [7]. Some efforts have been made to incorporate SVM notions in HK classifers. In [8], a regularization parameter C is introduced to control the tradeoff between the margin maximization and the training error rate. It has been shown that this method can yield reasonable performances in a short amount of calculation time. Yet this method suffers from the same problem as most of the regularized learning algorithms: the tuning of the regularization parameter which is commonly carried out with many tests, much time loss and a non-exhaustive scanning of the parameter values. Some authors proposed other methods including evolutionary tuning [9] or gradient-based approaches [10] for the tuning of SVM hyperparameters.

In this paper we introduce early stopping as a new regularization technique for HK learning called Ho–Kashyap with Early Stopping (HKES). The proposed method allows to automatically tune the parameter which in that case becomes the number of iterations before the training is stopped. More precisely, since the training is not actually stopped, this is the number of iterations retained for the final classifier. This procedure provides a very fast approximation of the best performance one can expect with a linear classifier on a given data set. This can give a good idea of the linear separability of the data. In some situations when the problem involves linearly separable data, this avoids to make the solution more complex by introducing non linear kernels. Such situations can occur in real-world applications, see for instance [11]. Nevertheless, choosing to tune a number of iterations as a hyperparameter leads to other problems like its sensitivity to the size of the training set as we will see in the numerical experiments.

We start in Sect. 2 with the description of 7 linear classifiers: Ho–Kashyap learning rule (HK), Modified Ho–Kashyap (MHKS and MHKA), Ho–Kashyap with Early Stopping (HKES) and with Absolute approximation of the errors (HKESA), Support Vector Machine (SVM) and finally, Lagrangian Support Vector Machine (LSVM). The methods MHKA, MHKS, SVM and LSVM appear for a comparison of the performances of HKES and HKESA through numerical experiments in Sect. 3. In this Section a discussion on the efficiency of the early stopping (HKES) and the problems that it implies is also presented, before concluding in Sect. 4.

Notations: all vectors are column vectors unless mentioned otherwise. I stands for the identity matrix. $1_{N \times 1}$ is the N-dimensional column vector with all components set to 1 and $1_{N \times N}$ is the $(N \times N)$-matrix with all components set to 1.

2 Linear classifiers

In a linear binary classification problem, the task is to find a hyperplane separating 2 classes. Let $(x_i, \; y_i)_{1 \leq i \leq N}$ be a set of training samples with $x_i \in \mathbb{R}^p$ belonging to a class labeled by $y_i \in \{+1, \; -1\}$. The decision function of a linear classifier is:

$$f(x) = \text{sign}\left(x^t w + w_0\right) \qquad (1)$$

where $(w \in \mathbb{R}^p, \; w_0 \in \mathbb{R})$ are the parameters (or weights) of the separating hyperplane to be estimated. If all the training samples are correctly separated, then:

$$y_i \left(x_i^t w + w_0\right) > 0, \quad i = 1, \ldots, N \; . \qquad (2)$$

2.1 Ho–Kashyap learning rule (HK)

Defining a set of N $(p+1)$-row vectors X_i:

$$X_i = \begin{cases} (x_i^t, +1) & \text{, if } y_i = +1 \\ (-x_i^t, -1) & \text{, if } y_i = -1 \end{cases} \qquad (3)$$

and a $(p+1)$-dimensional augmented weight vector $W = (w^t, w_0)^t$ allows to write (2): $X_i W > 0$, $i = 1, \ldots, N$. Then piling up the X_i, for $i = 1, \ldots, N$, in a $(N \times (p+1))$-matrix X gives:

$$XW > 0 \; . \qquad (4)$$

Let B be the "margin" vector with b_i as components. Equation (4) can be rewritten as:

$$XW = B$$
$$\text{subject to:} \quad B > 0 \; . \qquad (5)$$

The Ho–Kashyap learning rule [7] solves (5) by minimizing a least squares criterion. The problem is thus to minimize the square of the ℓ_2-norm of the errors:

$$J(W, B) = (XW - B)^t (XW - B)$$
$$\text{subject to:} \qquad B > 0 \; . \qquad (6)$$

The "margin" vector B is first initialized to B_0 with all components set to a small positive value b_0. At each step k, the weight vector W_k is deduced from B_k by:

$$W_k = (X^t X)^{-1} X^t B_k = X^\dagger B_k \qquad (7)$$

where $X^\dagger$ stands for the pseudo-inverse of X. Then a gradient descent is used to compute a new estimate of the margin vector. Defining the error vector

$E_k = XW_k - B_k$, the gradient of $J(W, B)$ with respect to B is $\nabla_B J(W, B)_k = -2E_k$. In order to satisfy the constraints $b_i > 0$, the positive components of $\nabla_B J(W, B)_k$, which are the negative components of E_k, are set to 0, preventing b_i to decrease and become negative. Thus $-(E_k + |E_k|)$ is used instead of $-2E_k$ and the iterative scheme is as follows:

$$B_{k+1} = B_k + \mu\,(E_k + |E_k|) \tag{8}$$

with μ a positive learning rate.

It can be shown [12] that this procedure converges in a finite number of steps $\forall\,\mu,\ 0 < \mu < 1$, to 0 in the separable case, to a non-zero value otherwise. This makes the tuning of μ not critical. In practice, one needs to define a termination criterion. This criterion is as follows:

$$|J(W, B)_k - J(W, B)_{k-1}| < \varepsilon\ . \tag{9}$$

2.2 Modified Ho–Kahsyap learning rule(MHKA-MHKS)

In [8], a hyperparameter C is introduced to tune the tradeoff between model complexity and training error. Adding a margin definition in (4): $XW \geq 1_{N \times 1}$, and a matrix D containing the reliability attached to each pattern, it comes to the Modified Ho–Kashyap classifier which is trained by minimizing the following criterion:

$$J(W, B) = (XW - 1_{N \times 1} - B)^t D(XW - 1_{N \times 1} - B) + Cw^t w\ . \tag{10}$$

The procedure remains the same except for (7), which becomes:

$$W_k = \left[(X^t D_k X + C\tilde{I})^{-1} X^t \right] D_k (B_k + 1_{N \times 1}) \tag{11}$$

where $\tilde{I}$ is the identity matrix with the last diagonal element set to 0, and the error vector which is now $E_k = XW_k - 1_{N \times 1} - B_k$.

For the ℓ_2-norm criterion, D is set to the identity matrix and remains constant, $D_k = I,\ \forall\ k$, allowing the pseudo-inverse in (11) to be calculated only once. The corresponding algorithm is termed Modified Ho–Kashyap with Squared approximation of the misclassification errors (MHKS).

The same author [8] proposes another scheme to increase the robustness to outliers: Modified Ho–Kashyap with Absolute approximation of the misclassification errors (MHKA). Instead of the ℓ_2-norm of the misclassification errors, the ℓ_1-norm is used as in the SVM algorithm with box constraints (see Sect. 2.5). An easy way to obtain this approximation is to let the diagonal components of D_k be $d_i = 1/|e_i|,\ i = 1, \ldots, N$, where e_i is the ith component of

the error vector E_k. Apart from the calculation of D_k at each iteration, the rest of the procedure remains unchanged. This algorithm is very similar to the Iteratively Reweighted Least Squares (IRLS) but differs by incorporating the modification of the vector B.

A major inconvenient of these methods is the tuning of C which is usually obtained by testing many values of C. Each test implies to run the learning process to its end and to estimate the resulting generalization error (GE). In case of a Leave One Out (LOO) estimate of GE, it becomes almost intractable in terms of calculation time. Therefore, k-fold cross-validation is often used instead of LOO.

2.3 Introducing early stopping (HKES)

Another way to avoid overfitting is early stopping, i.e. stopping the learning process before it reaches the optimal solution on the training set. Early stopping has been studied when applied on linear neural networks backpropagation in [13–15]. It was linked to regularization in [16] and recent works [17] showed that a margin perceptron with early stopping is equivalent to a soft-margin SVM.

We will show how one can incorporate early stopping in the original HK learning rule with automatic parameter tuning.

Let define k^* so that the algorithm has reached the convergence for $k = k^*$ iterations. Now, the aim is to find $\hat{k} < k^*$ for which GE is minimum. A very fast implementation of this task is to calculate an estimate GE_k of GE on an independent validation set at each iteration k, for $k = 1, \ldots, k^*$, during the learning process. Then $\hat{k}$ is given by:

$$\hat{k} = \operatorname*{argmin}_{k=1,\ldots,k^*}(GE_k) \; . \tag{12}$$

Thus, the process is executed only once, giving both $\hat{k}$ for GE minimum and the corresponding weight vector $\hat{W} = W_{\hat{k}}$. No other parameter has to be tuned. Indeed, a change in μ leads to a change in $\hat{k}$ which is automatically tuned.

Another common method for early stopping is to stop the learning as soon as GE_k is rising. But this method does not always yield the best minimum of GE since GE can have local minima (see for instance [18]). That is why we prefer to test all the generalization error GE_k estimates.

Compared to standard regularization (with C) used by MHKA and MHKS [8], this method is faster since only one learning is required. On the contrary,

to tune C, one should run the learning process for each tested value of C.

2.4 ℓ_1-norm for HKES (HKESA)

In order to gain in robustness to outliers, the ℓ_1-norm can be chosen for the criterion of the HKES algorithm. To obtain this, the same method as in Sect. 2.2 can be applied. A diagonal weighting matrix D is introduced in the calculation of the criterion (6):

$$J(W, B) = (XW - B)^t D(XW - B). \tag{13}$$

and W_k is computed in an Iteratively Reweighted Least Squares (IRLS) scheme by:

$$W_k = \left[(X^t D_k X)^{-1} X^t \right] D_k B_k \tag{14}$$

where the diagonal components of D_k are equal to $d_i = 1/|e_i|$, $i = 1, \ldots, N$, like in the MHKA method. The procedure is carried out for regularization by early stopping as explained for HKES in Sect. 2.3.

2.5   Support Vector Machine (SVM)

The Support Vector Machines [1,3] rely on the margin maximization. But to deal with noisy data, soft margin SVMs relax this constraint and authorize training errors. This is done by adding a regularization hyperparameter C that tunes the tradeoff between the model complexity and the training error. Note that this hyperparameter plays an inverse role compared to C in (10). As the margin equals $2/\|w\|$, the soft margin SVM problem becomes the minimization of:

$$J(w, w_0, \xi) = \frac{1}{2}w^t w + C\xi^t 1_{N\times 1}$$

$$\text{subject to: } Y(\tilde{X}w + w_0 1_{N\times 1}) \geq 1_{N\times 1} - \xi \tag{15}$$

$$\xi \geq 0$$

with $\tilde{X} = [x_1 \ldots x_N]^t$ the matrix of the non-augmented examples vectors, $Y = diag(y_i)$ the outputs matrix and ξ a vector of slack variables. It must be noticed that here, the ℓ_1-norm of the slack variables is used. The dual of this problem is [3]:

$$\max L_D = -\frac{1}{2}\alpha^t Y \tilde{X} \tilde{X}^t Y \alpha + 1_{N\times 1}^t \alpha$$

$$\text{subject to: } 0 \leq \alpha \leq C1_{N\times 1} \tag{16}$$

$$1_{N\times 1}^t Y \alpha = 0$$

which is a Quadratic Programming (QP) problem involving the so called box constraints: $0 \leq \alpha \leq C1_{N \times 1}$, where α is the N-vector of the Lagrange multipliers. The classifier parameters can be recovered from the solution $\hat{\alpha}$ of (16) by:

$$\hat{w} = \tilde{X}^t Y \hat{\alpha}, \text{ and } \hat{w}_0 = y_{i*} - x_{i*}{}^t w \tag{17}$$

where i^* is chosen such as $0 < \hat{\alpha}_{i*} < C$.

2.6 Lagrangian Support Vector Machine (LSVM)

The Lagrangian Support Vector Machine (LSVM) [6] is a much faster implementation of SVM (at least in the linear case) thanks to a reformulation of the problem that leads to the minimization of an unconstrained differentiable convex function. Thus the problem becomes solvable by a simple algorithm that requires only standard native MATLAB commands without any optimization tools such as linear or quadratic programming solvers. This derives from the choice of the ℓ_2-norm of the errors in the regularization term of the soft margin SVM minimization problem. However, the use of the ℓ_2-norm decreases the robustness to outliers. This choice together with the introduction of w_0 in the criterion leads to the minimization of:

$$J(w, w_0, \xi) = \frac{1}{2} \left(w^t w + w_0^2 \right) + \frac{C}{2} \xi^t \xi$$
$$\text{subject to: } Y(\tilde{X}w + w_0 1_{N \times 1}) \geq 1_{N \times 1} - \xi \tag{18}$$

with $\tilde{X}$ and $Y = diag(y_i)$ defined as in (15). The constraints $\xi_i \geq 0$ are not present because the choice of the ℓ_2-norm makes them redundant. The dual of this problem is:

$$\min L_D = \frac{1}{2} \alpha^t \left(Y(\tilde{X}\tilde{X}^t + 1_{N \times N})Y + \frac{I}{C} \right) \alpha - 1_{N \times 1}^t \alpha$$
$$\text{subject to: } \alpha \geq 0 . \tag{19}$$

The classifier parameters can be recovered from the solution $\hat{\alpha}$ of (19) by:

$$\hat{w} = \tilde{X}^t Y \hat{\alpha}, \text{ and } \hat{w}_0 = -1_{N \times 1}^t Y \hat{\alpha} . \tag{20}$$

Defining two matrices $H = Y[\tilde{X} \; -1_{N \times 1}]$ and $Q = \frac{I}{C} + HH^t$, the dual problem becomes:

$$\min_{0 \leq \alpha \in \mathbb{R}^N} f(\alpha) = \frac{1}{2} \alpha^t Q \alpha - 1_{N \times 1}^t \alpha . \tag{21}$$

In [6], an iterative algorithm is proposed to solve (21):

$$\alpha_{k+1} = Q^{-1} \left[1_{N \times 1} + \frac{1}{2} \left(Q\alpha_k - 1_{N \times 1} - \mu\alpha_k + |Q\alpha_k - 1_{N \times 1} - \mu\alpha_k| \right) \right] \tag{22}$$

and its convergence proof is given $\forall \mu, 0 < \mu < 2/C$.

3   Numerical experiments

In this section, we compare the previously described algorithms: Ho–Kashyap (HK) learning rule (see Sect. 2.1), Ho–Kashyap with Early Stopping and Squared (HKES, see Sect. 2.3) or Absolute (HKESA, see Sect. 2.4) approximation of the misclassification errors, Modified Ho–Kashyap with Squared (MHKS) or Absolute (MHKA) approximation of the misclassification errors [8] (see Sect. 2.2), Support Vector Machine (SVM) with ℓ_1-norm soft margin, and Lagrangian Support Vector Machine (LSVM) [6] (see Sect. 2.5 and 2.6) that uses the ℓ_2-norm. In order to evaluate the methods, 9 standard benchmark data sets from the IDA repository [19] are used. Each data set includes 100 predefined splits of the data[1] into training and test sets, thus allowing to calculate the average and standard deviation of the test error over 100 runs. More details and results of non-linear kernel methods on these data sets can be found in [20].

The following experiments expose two methods for the tuning of hyperparameters. In the first set of experiments, the training data are split into two sets: one training set and one validation set used for the hyperparameters tuning. In the second set of experiments, the test set on which the generalization error is estimated is used to tune the hyperparameters.

Another common procedure for the tuning of hyperparameters is cross-validation, either Leave One Out (LOO) or k-fold. But this procedure is not adapted to the HKES algorithm. The number of iterations tuned by HKES is very sensitive to the size of the training set and to the data. For instance, by estimating with 5-fold cross-validation the best number of iterations on folds of size of 4/5 of the training set, one would get a number of iterations that is irrelevant for another fold or for a training set of full size. This shows that the number of iterations is not a "good" hyperparameter in the sense that its value has no proper meaning and is affected by irrelevant factors as the number of examples. The estimation of the best number of iterations becomes thus impossible with cross-validation procedures.

---

[1] except for Slice and Image data sets that contain only 20 splits

8

3.1 Experiments setup

In all the experiments, and for all the Ho–Kashyap based classifiers, B is first initialized to B_0 with all components set to $b_0 = 10^{-6}$. Tests have been made and showed that the choice of b_0 does not affect the results if the termination criterion ε (9) is set in accordance with b_0. A heuristic choice that appears convenient is $\varepsilon = 0.5b_0^2$. For HKES and HKESA, the early stopping parameter $\hat{k}$ is evaluated as described in Sect. 2.3. Figure 1 shows an example of the evolution of the error on the validation set (the validation error) with respect to the number of iterations. The shape of this curve depends much on the size of the training set and the data themselves. Besides, tests have been made and showed that the choice of the learning rate μ does not influence the results or not for more than 0.05% (in terms of test error rate) over the 100 runs. It can thus be claimed that this method requires no parameter tuning. In our experiments, μ is set to 0.4.

To determine the regularization parameter C (for MHKA, MHKS, SVM and LSVM), a grid search is performed in two steps. First, an exponentially growing sequence of C is tested ($C = 2^{-10}, 2^{-8}, \ldots, 2^{10}$). Then 9 new testing values are taken in the neighborhood of the best value 2^p ($C \in [2^{p-1}, 2^{p+1}]$) to perform a fine tuning. This rather usual procedure allows to find a good approximation of the best C by scanning a wide range of values in a fixed and rather small number of tests (20 tested values). Figure 2 shows an example of the validation error for different values of C for the LSVM classifier.

3.2 Experiments with a validation set

In this set of experiments, the data is split into 3 independent sets (training set, validation set and test set). The classifier is trained on the training set while the validation set is used to tune the hyperparameters. Then the generalization error GE is estimated on the test set. One drawback of this method is to reduce the size of the training set to form the validation set. Moreover, the optimal ratio between the sizes of the training and the validation sets is not easily determined. The data sets are preprocessed as follows. Each predefined training set is divided into a smaller training set (2/3 of the original size) and a validation set. The 100 predefined test sets are kept unchanged.

Table 1 compares the average test error rates and the standard deviations over 100 runs respectively for HK, HKES, HKESA, MHKS, MHKA, LSVM and SVM. All results are in percentages.

It must be first noticed that the difference between the best and the worst result is never very large. In 5 cases out of 9, the performances of the HKES algo-

Table 1. Test error rates on benchmark data sets for HK, HKES, HKESA, MHKS, MHKA, LSVM and SVM tuned on a validation set. Boldface indicates the lowest error rate. *Italic* is used to highlight the cases where the error rate of HKES is higher than the error rate of HK. An asterisk (*) marks the error rates not significantly different from the best error rate at significance level $\alpha = 5\%$.

| Data set	HK	HKES	HKESA	MHKS	MHKA	LSVM (ℓ_2)	SVM (ℓ_1)
Diabetis	*23.69 ± 1.81	*23.75 ± 1.79	**23.69 ± 1.77**	24.80 ± 2.46	26.14 ± 4.61	*24.20 ± 2.12	*23.98 ± 2.83
Heart	18.20 ± 3.92	*17.17 ± 3.33	*17.07 ± 3.39	*16.83 ± 3.42	*16.43 ± 3.52	**16.28 ± 3.49**	18.36 ± 5.94
Breast-cancer	*28.35 ± 5.19	*28.19 ± 5.08	*28.41 ± 4.98	*28.21 ± 5.04	*28.35 ± 4.64	**27.36 ± 4.94**	29.13 ± 4.78
Splice	17.02 ± 0.82	16.93 ± 0.60	17.11 ± 0.71	16.90 ± 0.72	**16.44 ± 0.65**	17.07 ± 0.75	*16.73 ± 0.77
Image	17.33 ± 0.90	16.99 ± 1.06	17.25 ± 1.11	17.25 ± 1.35	19.07 ± 2.20	17.37 ± 1.23	**15.41 ± 0.83**
German	*24.27 ± 2.30	*24.39 ± 2.19	24.92 ± 2.49	*24.79 ± 2.32	27.23 ± 2.73	**24.26 ± 2.16**	*24.90 ± 2.77
Flare-solar	33.68 ± 1.81	33.65 ± 1.69	36.20 ± 13.2	35.98 ± 5.24	34.55 ± 3.95	33.81 ± 2.00	**32.58 ± 2.49**
Thyroid	*10.96 ± 2.89	*11.89 ± 3.50*	15.03 ± 3.44	*11.04 ± 2.95	15.40 ± 3.23	11.77 ± 3.41	**10.39 ± 2.86**
Titanic	**22.81 ± 1.48**	**22.81 ± 1.48**	*22.82 ± 1.52	23.75 ± 3.23	23.66 ± 3.09	*22.85 ± 1.35	25.07 ± 4.27
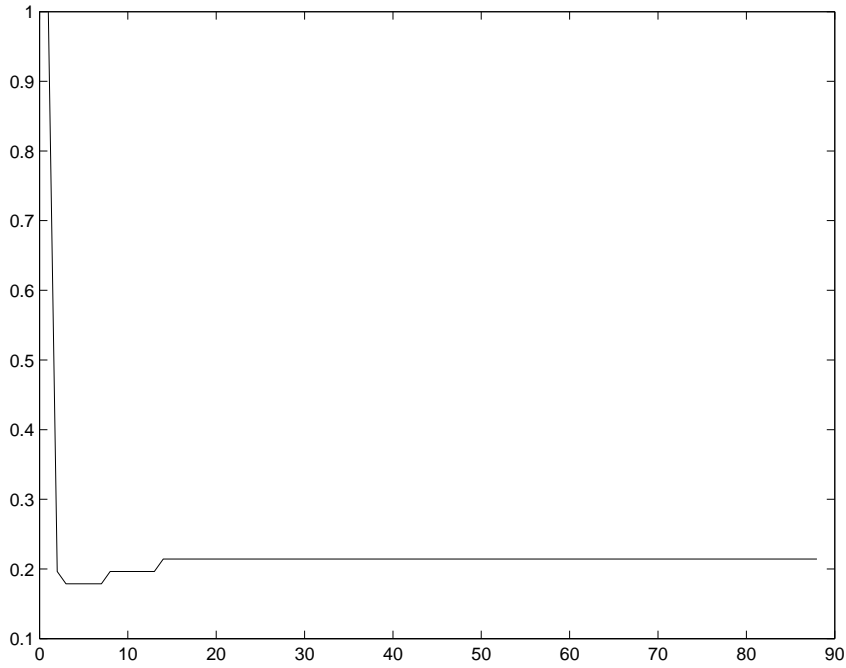
Fig. 1. An example of the evolution of the validation error during the training of HKES.
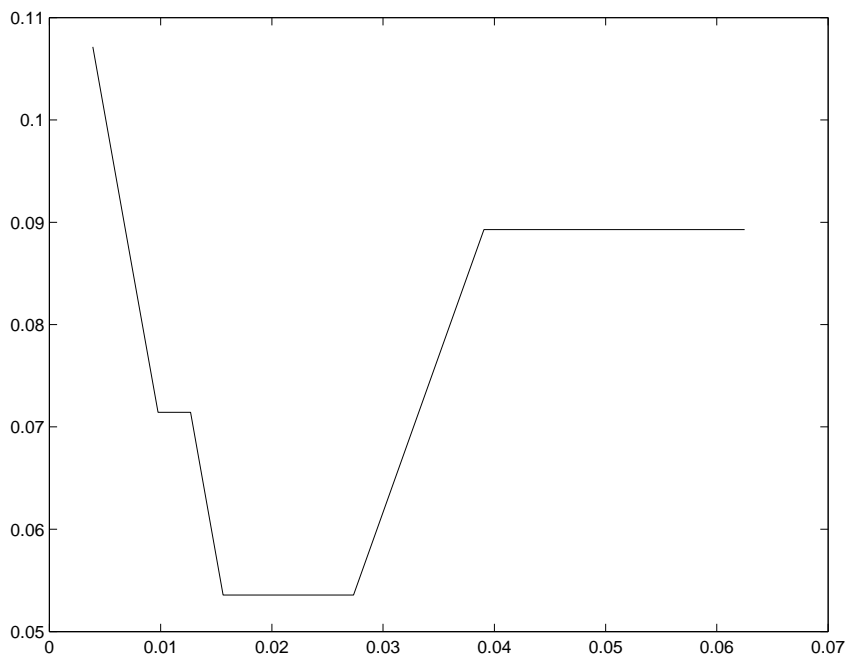

Fig. 2. An example of the evolution of the validation error for different values of C during the fine tuning for the LSVM classifier.

rithm are not significantly different from those of the best classifier. Nonetheless, it is clear that the margin maximization of both SVM and LSVM makes them better in generalization, though MHKS and MHKA classifiers tend also to maximize the margin. The effect of the ℓ_1-norm as a more outliers-robust

11

loss function can be seen between HKES, MHKS, LSVM and HKESA, MHKA, SVM respectively. For instance, on the diabetis data set, the best classifier (HKESA) uses the ℓ_1-norm and for this data set, all the ℓ_1-norm classifiers provide better performances than the corresponding ℓ_2-norm classifiers. So this data set might contain a certain amount of outliers. It must be also noticed that on three data sets (image, splice and thyroid) the results can be highly enhanced by using non linear classifiers as in [20]. In these cases, a simple linear hyperplane is not sufficient to separate the data with satisfying results.

In order to evaluate the effect of the early stopping, a comparison is performed between Ho–Kashyap with Early Stopping (HKES) and the same algorithm without any tuning of the number of iterations or early stopping (HK). For the data set titanic, these two classifiers obtain the same performance which is also the best performance for this data set. For 5 data sets, the early stopping allows to improve the results and the average gain between HKES and HK is 0.33%. For the 3 remaining data sets (highlighted in *italic* in Table 1), the test error rates of HKES are higher than those of HK and the average loss is 0.37%. This might be explained by the overfitting on the validation set. Nevertheless, the gain over all data sets is +0.54%. So the results of the early stopping for the HK algorithm are not very clear. The tunning of a very low level hyperparameter (the number of iterations) is easy to implement and fast (see Table 2 and the following paragraph) but can also be tricky as it can sometimes decrease the performances when applied on a validation set.

Table 2
Calculation times in seconds. The number of runs appear next to the data set name.

Data set	HKES	HKESA	MHKS	MHKA	LSVM	OSU-SVM
Diabetes (100)	**1**	58	47	814	86	360
Heart (100)	**4**	12	57	142	45	80
Breast-cancer (100)	**1**	12	11	214	34	77
Splice (20)	**11**	95	508	1340	424	2875
Image (20)	**10**	2760	718	3200	195	1990
German (100)	**12**	228	5000	3500	182	2090
Flare-solar (100)	**5**	460	843	3340	112	143
Thyroid (100)	24	**4**	215	41	75	41
Titanic (100)	**0.5**	5	6	35	5	8

Mean Ratio between LSVM and HKES times: 47

Table 2 compares calculation times for HKES, HKESA, MHKS, MHKA, SVM and LSVM. Computations are run on an AMD Athlon 2000+ with 256 Mb of

memory running Windows XP and MATLAB 6.5. The SVM implementation used is provided by the OSU SVM toolbox[2] written for MATLAB with the core in C++ that uses LIBSVM algorithm. The LSVM algorithm is entirely coded in MATLAB (the code can be found in [6]). The times appear in seconds for 100 runs. Each run includes the learning on the training set and the parameter tuning on the validation set. It must be noticed that for MHK(S-A), SVM and LSVM, the times depend strongly on the number of tested values for C. The HKES algorithm is about 50 times faster than the LSVM algorithm tuned over 20 values of C. The use of gradient-based methods for the tuning of SVM hyperparameters [10] could increase the speed. But even by testing only one value of C, the HKES algorithm is still faster than the LSVM algorithm partly because it ensures a complete scanning of the possible values for its hyperparameter $\hat{k}$ in only one training. The HKESA classifier benefits from the same advantage, but when ℓ_1-norm is used in a HK based algorithm, the pseudo-inverse $(X^t D_k X)^{-1} X^t$ has to be computed at each iteration. This explains the large difference in the calculation time between HKES and HKESA and between MHKS and MHKA.

3.3 Experiments without a validation set

The procedure used in this Section for the comparison of the performances of the classifiers is as follows. The 100 predefined splits between the training and test sets are kept unchanged. For each split the training is performed on the training set and the test set is used to tune the hyperparameters of the classifier. The estimation of GE is done on the test set too. This procedure can lead to a biased estimate of GE because the test set is no more independent. So the results do not stand for the exact performances of the classifiers but since the same procedure is used for all the classifiers, they can be considered as indicators for the comparison of the algorithms.

Table 3 compares the average test error rates and the standard deviations respectively for HK, HKES, HKESA, MHKA, MHKS, SVM and LSVM over 100 runs. All results are in percentages. In this case, early stopping on the HK based algorithms is, in 8 cases out of 9, not significantly different from the best classifier, whereas Modified Ho-Kashyap methods (MHKS and MHKA) that use C-regularization only achieve this on 6 out of 9 data sets. Moreover, the effect of early stopping for regularization can be seen by noting that the HK learning rule without early stopping is always significantly different from the best classifier except for the splice data set on which HKES gives the lowest test error rate.

---

[2] OSU SVM Classifier Matlab Toolbox, by Junshui Ma, Yi Zhao, and Stanley Ahalt. Downloadable from http://www.ece.osu.edu/~maj/osu_svm/ .

Table 3. Test error rates on benchmark data sets for HK, HKES, HKESA, MHKS, MHKA, LSVM and SVM tuned on the test set. Boldface indicates the lowest error rate. An asterisk (*) marks the error rates not significantly different from the best error rate at significance level $\alpha = 5\%$.

Dataset	HK	HKES	HKESA	MHKS	MHKA	LSVM (ℓ_2)	SVM (ℓ_1)
Diabetis	23.42 ± 1.80	$^*23.15 \pm 1.77$	$^*22.94 \pm 1.71$	$^*22.98 \pm 1.64$	$^*23.12 \pm 1.70$	$^*22.92 \pm 1.76$	$\mathbf{22.86 \pm 1.51}$
Heart	16.48 ± 2.83	$^*15.36 \pm 2.87$	$^*15.11 \pm 3.33$	$^*15.20 \pm 2.96$	$^*15.14 \pm 3.24$	$\mathbf{14.46 \pm 2.99}$	$^*14.60 \pm 3.04$
Breast-cancer	27.09 ± 4.80	26.86 ± 4.73	$^*25.82 \pm 4.59$	26.40 ± 4.72	26.57 ± 4.65	$\mathbf{24.64 \pm 4.54}$	26.16 ± 4.38
Splice	$^*16.17 \pm 0.72$	$\mathbf{15.90 \pm 0.67}$	$^*16.10 \pm 0.57$	$^*15.98 \pm 0.67$	$^*16.31 \pm 0.59$	$^*15.94 \pm 0.66$	$^*15.95 \pm 0.52$
Image	17.08 ± 0.92	16.37 ± 0.94	$^*15.72 \pm 0.91$	16.63 ± 0.86	18.02 ± 1.48	16.64 ± 0.87	$\mathbf{15.20 \pm 0.84}$
German	24.07 ± 2.15	23.84 ± 2.09	23.90 ± 1.91	23.67 ± 2.09	26.31 ± 2.85	$\mathbf{22.89 \pm 2.13}$	23.59 ± 2.23
Flare-solar	33.47 ± 1.50	33.31 ± 1.51	$^*32.67 \pm 1.60$	32.92 ± 1.60	$^*32.55 \pm 1.66$	$^*32.67 \pm 1.60$	$\mathbf{32.27 \pm 1.81}$
Thyroid	10.57 ± 2.35	$^*9.76 \pm 2.32$	14.61 ± 3.27	$^*9.97 \pm 2.30$	15.15 ± 3.12	$^*9.89 \pm 2.34$	$\mathbf{9.45 \pm 2.39}$
Titanic	22.68 ± 1.11	22.68 ± 1.10	$^*22.63 \pm 1.11$	$^*22.54 \pm 1.16$	$^*22.50 \pm 1.15$	$\mathbf{22.36 \pm 1.13}$	$^*22.60 \pm 1.11$

14

4   Conclusion

In this paper, we investigated the effect of early stopping as a regularization method for the Ho–Kashyap (HK) learning rule. Moreover, the choice of a more outliers-robust criterion (ℓ_1-norm) has been studied. Tests on benchmarks have been conducted to compare early stopping to the standard regularization (with the addition of a tradeoff parameter in the criterion) on both HK and SVM algorithms. All of these methods gave comparable results. But it is clear that Ho–Kashyap with Early Stopping (HKES) is much faster than the others partly because it requires no hyperparameter tuning. It must also be noticed that when no independent validation set is used, early stopping on the HK based algorithms gives, in 8 cases out of 9, results not significantly different from the ones obtained with the best classifier which, in most cases, is a SVM classifier. Modified Ho-Kashyap methods (MHKS and MHKA) that use standard regularization only achieve this on 6 out of 9 data sets. It must be nevertheless noticed that in this case, the estimation of the generalization error has been performed on the same test set used to tune the hyperparameters: the number of iterations for the proposed methods based on the HK learning rule (HKES and HKESA), the regularization parameter C for the reference methods based also on the HK learning rule (MHKS and MHKA) or on SVM framework. This estimation is common for all the methods, thus allowing a comparison, but can introduce a bias.

Regularization by early stopping is not adapted to other estimations of the generalization error, such as cross-validation procedures. Indeed, the number of iterations used for regularization is a too low level parameter which depends too closely on the number of data and the data themselves. When the performance is estimated on a test set independent from the validation set used to tune the hyperparameters, the early stopping tends sometimes to overfit on the validation set. The tuning of the early stopping (HKES) can in these cases lead to higher test error rates than without any tuning or early stopping (HK). The overfitting on the data used for the tuning explains also why HKES and HKESA give appreciable results when no independent validation set is used and that the test set is used for the tuning of the hyperparameters and for the comparison of the performances. This problem can occur in some applications when one has not enough data at hand and tends to divide the data only into a training set and a test set. The method that overfits on the test set will show the best results but will not necessarily provide the best classifier for the application.

5 Acknowledgements

References

[1] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, 1995.

[2] C. J. Burges, A tutorial on support vector machines for pattern recognition, Data mining and knowledge discovery 2 (1998) 121–167.

[3] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000.

[4] C. W. Hsu, C. J. Lin, A simple decomposition method for support vector machines, Machine Learning 46 (2002) 291–314.

[5] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, MIT Press, 1999, Ch. 11.

[6] O. L. Mangasarian, D. R. Musicant, Lagrangian support vector machines, Journal of Machine Learning Research 1 (2001) 161–177.

[7] E. Ho, R. L. Kashyap, An algorithm for linear inequalities and its applications, IEEE Trans. Electronic Computers 14 (1965) 683–688.

[8] J. Łęski, Ho–Kashyap classifier with generalization control, Pattern Recognition Letters 24 (2003) 2281–2290.

[9] F. Friedrichs, C. Igel, Evolutionary tuning of multiple SVM parameters, in: M. Verleysen (Ed.), 12th European Symposium on Artificial Neural Networks (ESANN), 2004, pp. 519–524.

[10] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (1-3) (2002) 131–159.

[11] F. Lauer, M. Bentoumi, G. Bloch, G. Millérioux, P. Aknin, Ho–kashyap with early stopping vs soft-margin SVM for linear classifiers – an application, in: Advances in Neural Networks - ISNN 2004, International Symposium on Neural Networks, Dalian, China.

[12] R. O. Duda, P. E. Hart, D. G. Stork, Pattern Classification, 2nd Edition, Wiley, 2000.

[13] P. Baldi, Y. Chauvin, Temporal evolution of generalization during learning in linear networks, Neural Communication 3 (1991) 589–603.

[14] R. Dodier, Geometry of early stopping in linear networks, in: D. S. Touretzky, M. C. Mozer, M. E. Hasselmo (Eds.), Advances in Neural Information Processing Systems, Vol. 8, The MIT Press, 1996, pp. 365–371.

[15] C. Wang, S. S. Venkatesh, J. S. Judd, Optimal stopping and effective machine complexity in learning, in: Neural Information Processing Systems (NIPS), 1993, pp. 303–310.

[16] J. Sjoberg, L. Ljung, Overtraining, regularization, and searching for minimum in neural networks, in: Preprints 4th IFAC Symp. on Adaptive Systems in Control and Signal Processing, 1992, pp. 669–674.

[17] R. Collobert, S. Bengio, Links between perceptrons, MLPs and SVMs, in: International Conference on Machine Learning, ICML, 2004.

[18] L. Prechelt, Early stopping – but when ?, Lectures Notes in Computer Science 1524 (1998) 55–69.

[19] Ida benchmark repository used in several boosting, KFD and SVM papers, http://ida.first.fraunhofer.de/projects/bench/.

[20] G. Rätsch, T. Onoda, K.-R. Müller, Soft margins for AdaBoost, Machine Learning 42 (3) (2001) 287–320, also NeuroCOLT Technical Report NC-TR-1998-021.