Optimal support vector selection for kernel perceptrons

Daniel García, Ana González, José R. Dorronsoro
Dpto. de Ingeniería Informática and Instituto de Ingeniería del Conocimiento
Universidad Autónoma de Madrid,
28049 Madrid, Spain

LEARNING 2006

Objectives

- A common problem with many SVM implementations is the relatively large number of support vectors they generate, as the cost of computing the output is $\mathrm{O}(|SV|)$ kernel operations
- In this work we will explore a new approach to obtain kernel perceptrons with a smaller number of support vectors
- In order to achieve this task, we start studying the SVM training method and how it performs the margin maximization

Support Vector Machines

- SVM goal: given a sample $\mathcal{S} = \{(X_i, y_i) : i = 1, \ldots, N\}$, where $y_i = \pm 1$, to construct a classifier $c(X) = W \cdot X + b$ with a maximum margin

$$m(W, b) = \min \left\{ \frac{y_i(W \cdot X_i + b)}{\|W\|} : i = 1, \ldots, N \right\}$$

- Alternatively, to solve

$$(W^*, b^*) = \operatorname{argmin}_{(W,b)} \|W\|^2$$

with (W, b) satisfying $y_i(W \cdot X_i + b) \geq 1$ for all i

Convex Hull Norm Minimization

- Writing $\tilde{\mathcal{S}} = \{y_i X_i : i = 1, \ldots, N\}$ and $C(\tilde{\mathcal{S}})$ its convex hull the maximum margin vector W^* verifies

$$W^* \quad = \quad \arg\min\{\|W\| : W \in C(\tilde{\mathcal{S}})\}$$

- Moreover, the optimal W^* verifies $m(W^*) = \|W^*\|$
- Thus, for any $W \in C(\mathcal{S})$ we have

$$m(W) \leq m(W^*) = \|W^*\| \leq \|W\|$$

- Hence, if $g(W) = \|W\| - m(W)$, then $0 = g(W^*) \leq g(W)$
- Therefore, minimizing $g(W)$ gives an optimum marging

Schlesinger–Kozinec (SK) Algorithm

- The SK algorithm seeks to minimize $g(W)$ in two steps
  - At step t it selects an X_l such that $l = \arg\min_i \{y_i W_t \cdot X_i\}$
  - Then it updates W_t as

$$W_t = (1 - \lambda^*) W_{t-1} + \lambda^* y_{l(t)} X_{l(t)}$$

with $\lambda^* = \arg\min_\lambda \{\|(1 - \lambda) W_{t-1} + \lambda X_{l(t)}\|\}$

- The above updates are applied even if all patterns are correctly classified and ensure $W_t \in C(\tilde{S})$
- The SK algorithm assures that

$$\|W_t\| \leq \|W_{t-1}\|$$

Kernel SK I

- By Mercer's theorem, a definite positive kernel $k(x, z)$ defines a non–linear mapping $X = (\phi(x), 1) = \Phi(x)$ such that

$$X \cdot Z = 1 + \phi(x) \cdot \phi(z) = 1 + k(x, z) = K(x, z)$$

- Writing $W_t = \sum_j \alpha_j^t y_j X_j = \sum_l \alpha_j^t y_j \Phi(x_j)$, the update of W_t can be written as

$$W_t = (1 - \lambda^*)\, W_{t-1} + \lambda^* y_l X_l = (1 - \lambda^*) \sum_j \alpha_j^{t-1} y_j X_j + \lambda^* y_l X_l$$

- Therefore $\alpha_j^t = (1 - \lambda^*)\, \alpha_j^{t-1} + \lambda^* \delta_{j\,l}$ and the cost of α updates is $\mathrm{O}(N)$

Kernel SK II

- To speed up the new pattern selection, we keep a margin vector $D_j^t = y_j W_t \cdot X_j, j = 1, \ldots, N$
- We choose the new pattern as

$$I = I(t) = \arg\min_i \left\{ D_i^{t-1} \right\}$$

λ^* can be obtained as

$$\lambda^* = \min\left(1, \frac{\|W_{t-1}\|^2 - D_I^{t-1}}{\|W_{t-1}\|^2 - 2D_I^{t-1} + \|X_I\|^2}\right)$$

and the margin updates are

$$D_j^t = (1 - \lambda^*) D_j^{t-1} + \lambda^* y_I y_j K(x_I, x_j)$$

while we have

$$\|W_t\|^2 = (1 - \lambda^*)^2 \|W_{t-1}\|^2 + 2(1 - \lambda^*)\lambda^* D_I^t + (\lambda^*)^2 \|X_I\|^2$$

Kernel SK III

- The cost of these operations is $\mathrm{O}(C\ N)$, with C the cost of a kernel computation.
- The cost of a T iteration SK pcp training becomes $\mathrm{O}(T\ C\ N)$, while memory requirements are just $\mathrm{O}(N)$.
- On the other hand, if the resulting perceptron is to be applied to a size S test set, the cost will then be $\mathrm{O}(|SV|\ C\ S)$, which may be similar to that of the whole training if $N \simeq |SV|$.

Support vector selection I

- W_t is defined in terms of the support vectors, as

$$W_t = \sum_j \alpha_j^t y_j X_j$$

- The fact that $W_t \in C(\tilde{\mathcal{S}})$ yields

$$\sum_j \alpha_j^t = 1,$$

  where α_j^t are non-negative coefficients.
- **Is it possible to see this as a probabilistic distribution?**

Support vector selection II

- The relative relevance of the support vectors, as given by the α_i coefficients, should be approximately the same. In other words, $\alpha_i \simeq 1/N$ is to be expected
- We can use this to set up a support vector removal method, which consists in removing those vectors with small α_i coefficients.
- First idea: rather than trying a direct computation to find out which vectors to remove, we will iteratively remove support vectors with "small" α_i, retraining the kernel perceptron using as training sample the remaning vectors.

Support vector removal method

- Let $\mathcal{S}_0 = \{(X_i^0, y_i^0)\}$ denote the initial training sample; we set an initial factor $\delta_0 < 1$ and after training finishes, we shall only keep in the next training sample $\mathcal{S}_1$ those (X_i^0, y_i^0) for which $\alpha_i \geq \tau_0 = \delta_0 / N$.
- We will iteratively apply this procedure with δ_k values that increase to 1, which will result in a decreasing sequence $\mathcal{S}_0 \supset \mathcal{S}_1 \supset \ldots$ of training sets
- Adittionaly, we can use the removed vectors as a validation set $\mathcal{V}_k = \mathcal{S}_0 \setminus \mathcal{S}_k$, which can be used to state a stop criterion. For instance, the iterative procedure would end when the validation accuracy got too low.

Numerical experiments I

- We shall work with six datasets from the UCI repository.
- We shall use the gaussian kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{\sigma^2}\right);$$

with an arbitrary value of $\sigma^2 = 50$

Numerical experiments II

- To guarantee linear separability we extend the projected patterns $X = \Phi(x)$ as $X_i' = \left(X_i, 0, \ldots, \frac{y_i}{2C}, \ldots, 0\right)$
- This requires to work with extended kernel

$$K'(x,z) = K(x,z) + \frac{1}{2C}\,\delta_{xz};$$

we shall use a common C value of 10

Numerical experiments III

- We shall perform a 10–fold cross–validation randomly splitting the datasets in 10 subsets
- For each dataset, we shall perform up to 20 retrainings with 10.000 epochs each, starting at $\delta_0 = 0.5$, and using a fixed increment $\nu = 0.025$ to thresholds $\tau_k = \delta_k / N$, with $\delta_k = \delta_0 + k\nu$.
- After each removal and retraining step we shall compute the validation set accuracy a_{vs} and the test set accuracy a_{ts}, averaged over the 10 subsets.
- The procedure ends when all retrainings are completed, or when $|a_{vs}^{k-1} - a_{vs}^{k}| - \sigma_{vs}^{k-1} > 0$, where σ_{vs}^{i} is the standard deviation of the validation set accuracy at step i.

Results

Dataset	#patterns	#initial SV	#final SV	%SV
heartdis	290	164.9	109.1	66.16
breastW	690	117.1	89.4	76.24
ionosphere	340	134.8	84.5	62.69
sonar	200	136.1	88.0	64.65
pima	760	542.0	416.0	76.75
thyroid	7190	1121.5	953.8	85.04

Table: Number of patterns, SV and SV reduction rates for six datasets.

Heart disease results

factor	validation acc.	test acc.	# supp. vect	stop c
0.0	–	0.793 ± 0.106	164.9 ± 6.9	–
0.650	0.991 ± 0.011	0.790 ± 0.106	116.2 ± 2.7	-0.001
0.675	0.988 ± 0.010	0.786 ± 0.104	112.1 ± 3.7	-0.005
0.700	0.980 ± 0.014	0.772 ± 0.104	109.1 ± 3.9	0.001

Table: Evolution of support vector reduction for Heart disease.

- Optimal factor = 0.700
- Support Vector Reduction = 66.16 %
- Final accuracy = 0.772

Wisconsin breast cancer results

factor	validation acc.	test acc.	# supp. vect	stop c
0.0	–	0.958 ± 0.021	117.1 ± 11.2	–
0.925	1.000 ± 0.000	0.959 ± 0.022	90.5 ± 5.2	0.000
0.950	1.000 ± 0.000	0.961 ± 0.022	89.8 ± 5.5	0.000
0.975	1.000 ± 0.000	0.961 ± 0.022	89.4 ± 5.2	0.000

Table: Evolution of support vector reduction for Wisconsin breast cancer.

- Optimal factor $= 0.975$
- Support Vector Reduction $= 76.24$ %
- Final accuracy $= 0.961$

Ionosphere results

factor	validation acc.	test acc.	# supp. vect	stop c
0.0	–	0.918 ± 0.037	134.8 ± 7.5	–
0.600	0.999 ± 0.004	0.915 ± 0.038	87.6 ± 3.2	-0.001
0.625	0.997 ± 0.006	0.915 ± 0.038	86.4 ± 3.2	-0.002
0.650	0.992 ± 0.014	0.918 ± 0.032	84.5 ± 3.4	0.001

Table: Evolution of support vector reduction for Ionosphere.

- Optimal factor = 0.650
- Support Vector Reduction = 62.69 %
- Final accuracy = 0.918

Pima indian diabetes results

factor	validation acc.	test acc.	# supp. vect	stop c
0.0	–	0.775 ± 0.072	542.0 ± 9.2	–
0.525	0.998 ± 0.004	0.767 ± 0.070	448.8 ± 7.4	-0.004
0.550	0.995 ± 0.004	0.764 ± 0.074	432.3 ± 7.2	-0.001
0.575	0.991 ± 0.005	0.770 ± 0.067	416.0 ± 8.6	0.001

Table: Evolution of support vector reduction for Pima indian diabetes.

- Optimal factor = 0.575
- Support Vector Reduction = 76.75 %
- Final accuracy = 0.770

Heart disease evolution

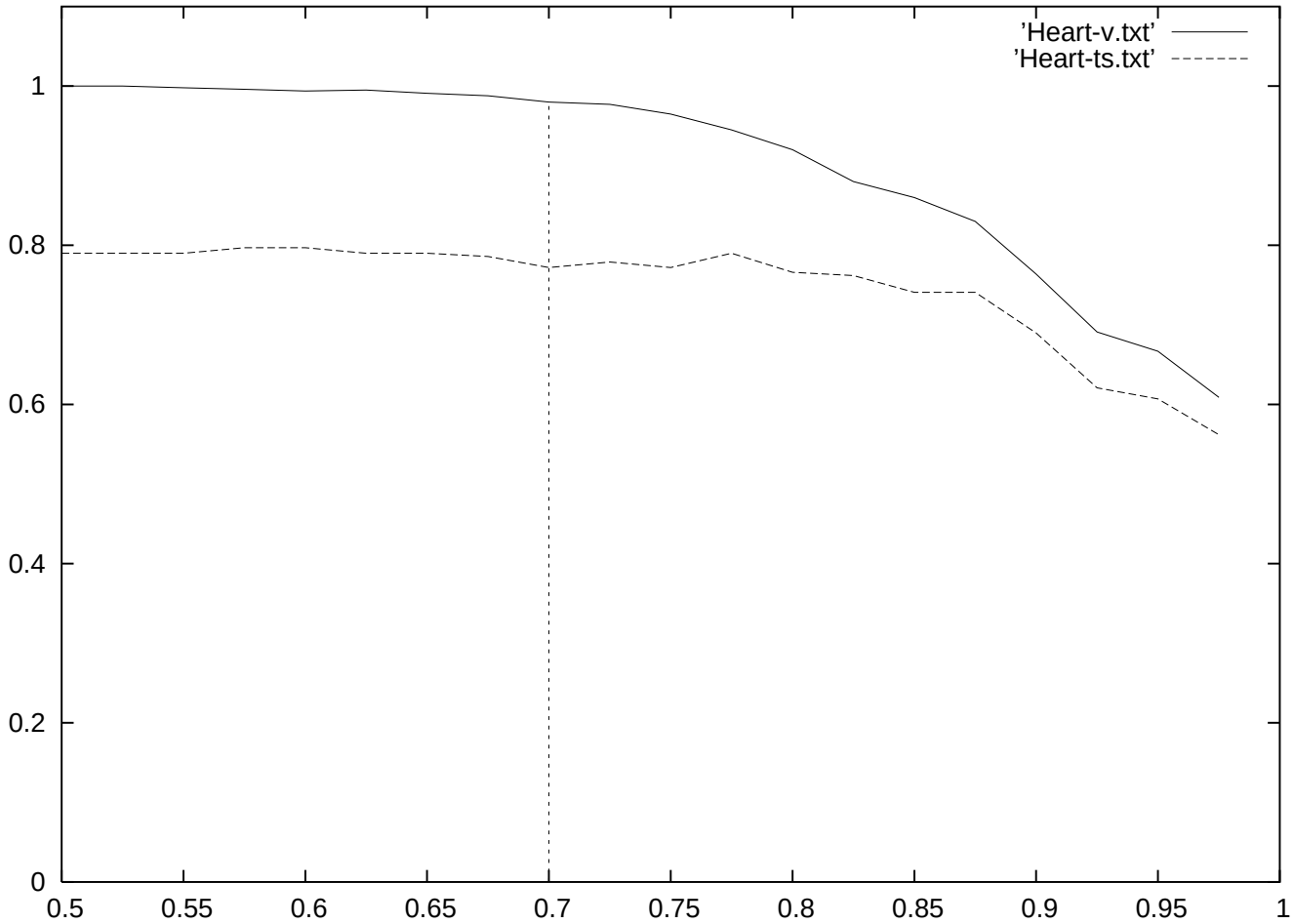

Figure: Accuracy evolution in validation (upper curve) and test (lower curve) for the Heart disease dataset.

Wisconsin breast cancer disease evolution

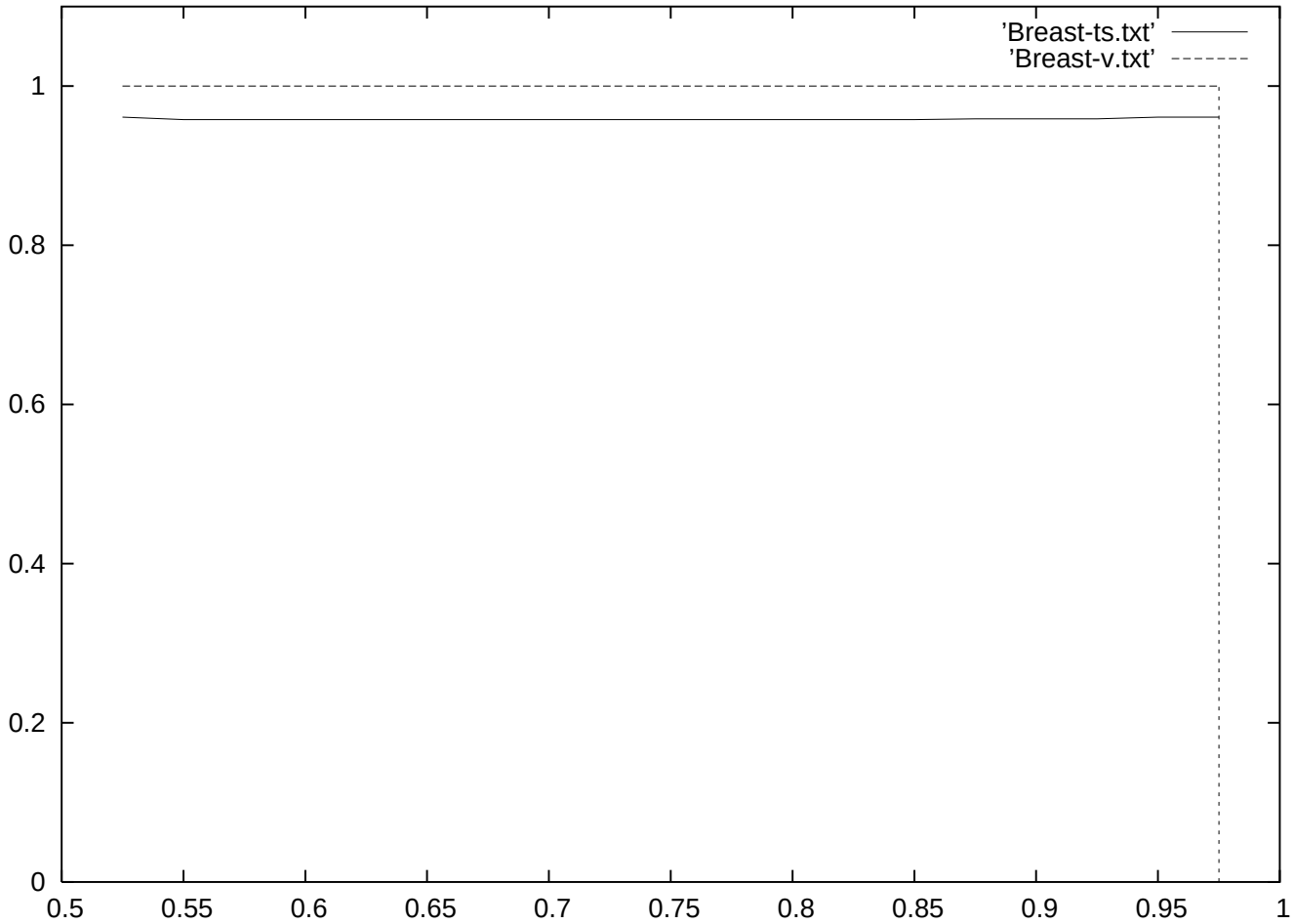

Figure: Accuracy evolution in validation (upper curve) and test (lower curve) for the Wisconsin breast cancer dataset.

Ionosphere evolution

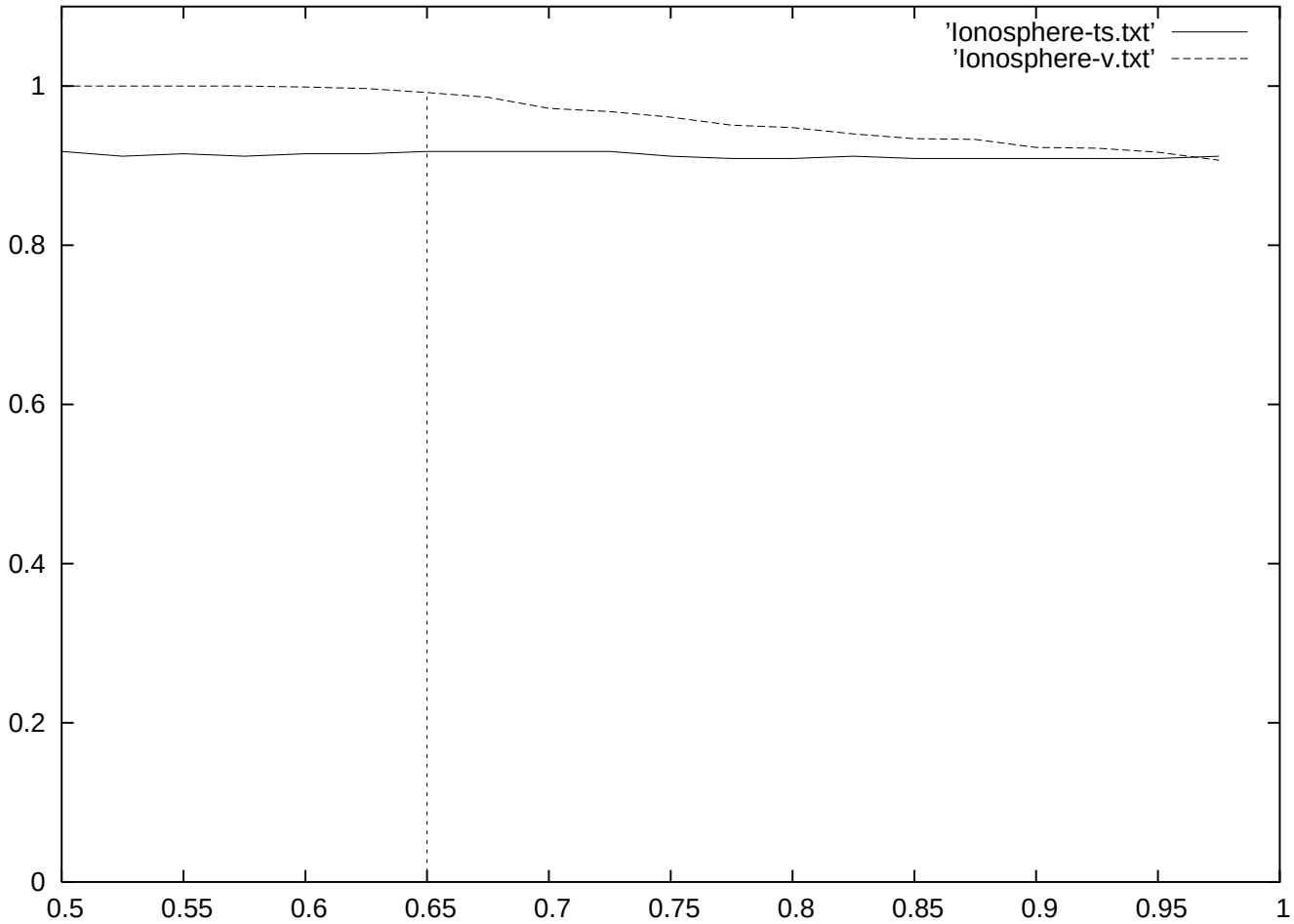

Figure: Accuracy evolution in validation (upper curve) and test (lower curve) for the Ionosphere dataset.

Pima indian diabetes evolution

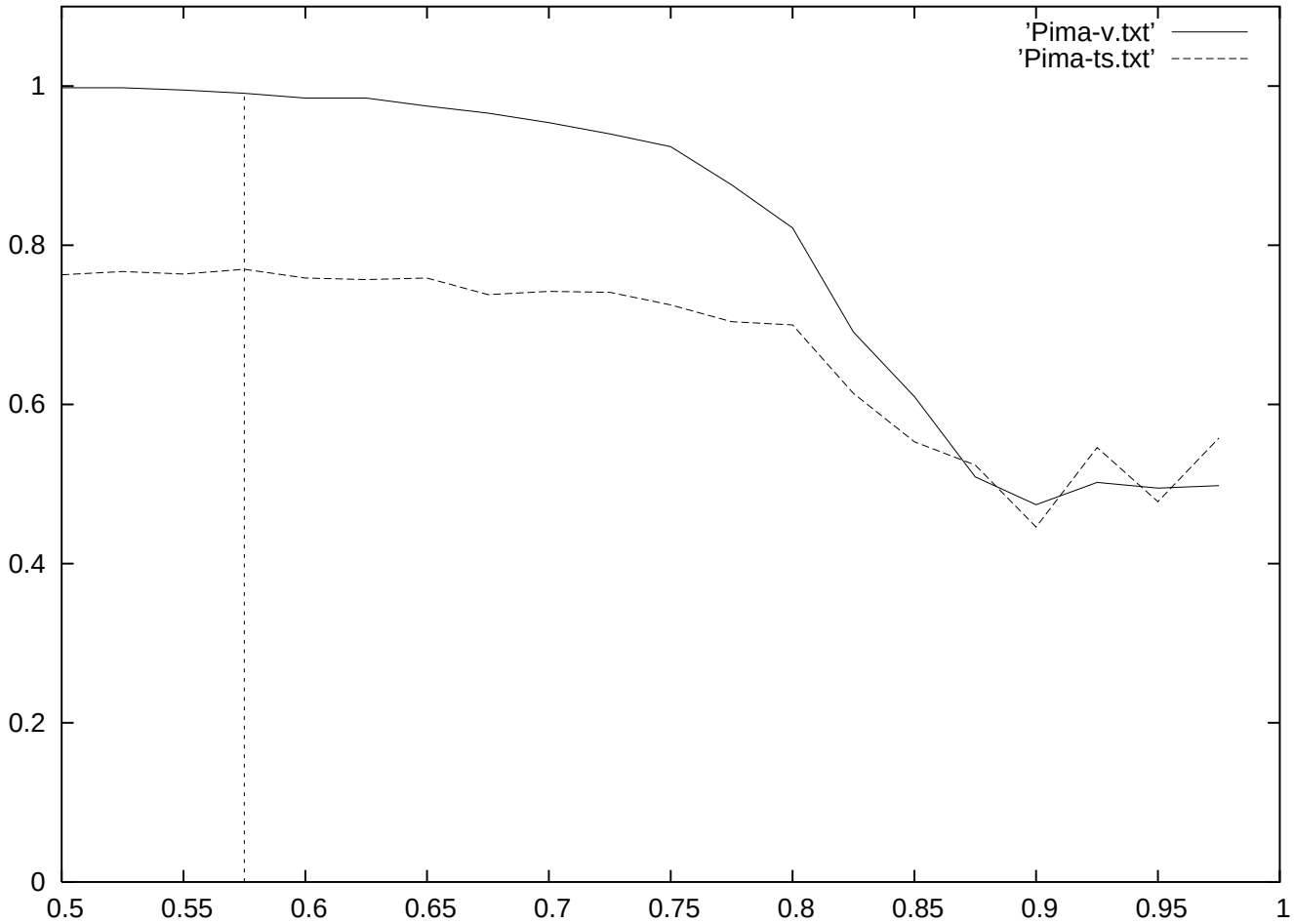

Figure: Accuracy evolution in validation (upper curve) and test (lower curve) for the Pima indian diabetes dataset.

Conclusions

- A common problem of kernel classifier construction methods is the high number of final support vectors they must use, resulting in a very high cost of new patterns classification.
- We have shown that the number of final support vectors can considerably be reduced while retaining a good classification performance.
- The work presented here has to be seen as being of an exploratory nature.
- Future work:
  - To look for a better stopping criterion.
  - To investigate other methods in the convex hull setting, such as "budget algorithms", in order to minimize the number of support vectors