Ensembles of Regularized Least Squares Classifiers for high dimensional Problems

written by Kari Torkkola and Eugene Tuv

Theodor Mader
tmader@student.ethz.ch

ETH Zurich, Switzerland

2th February 2006

Overview

The Setting

Problem Definition

- What we have: set of m patterns, consisting of features and labels $(\mathbf{x}_i, y_i)_{i=1}^m \quad \in \{\mathcal{X}^n \times \{-1, 1\}\}$
- What we want: find a function $f : \mathcal{X}^n \rightarrow \{-1, 1\}$ that predicts the label for a *new (never seen)* pattern.

Some Fundamental Problems:

- What is the optimal f?
- Feature space is usually very high dimensional
- Features might contain noise

Particular Setting

A linear decision function

For the space $\mathcal{F}$ of possible decision functions we coose all functions of the form

$$f(\mathbf{x}) = \sum_{j=1}^{n} \omega_j x_j = \omega \bullet \mathbf{x}$$

where x_j denotes the j-th component of pattern $\mathbf{x}$, and ω is a weight vector.

Particular Setting

A linear decision function

For the space $\mathcal{F}$ of possible decision functions we coose all functions of the form

$$f(\mathbf{x}) = \sum_{j=1}^{n} \omega_j x_j = \omega \bullet \mathbf{x}$$

where x_j denotes the j-th component of pattern $\mathbf{x}$, and ω is a weight vector.

The loss function

Our loss criterion will be a quadratic loss function:

$$L(f(\mathbf{x}), y) = (y - f(\mathbf{x}))^2$$

and we also introduce a regularisation term $||\omega||^2$

Optimisation Problem (1/2)

This leads us to the optimisation problem: Find ω such that the overall loss for all patterns $\mathbf{x}_i$

$$\frac{1}{m}\sum_{i=1}^{m}(\omega \bullet \mathbf{x}_i - y_i)^2 - \lambda||\omega||^2$$

is minimal. Putting all Patterns $\mathbf{x}_i$ into a Matrix $\mathbf{X}$ such that each pattern corresponds to a row in $\mathbf{X}$, and all labels into a vector $\mathbf{y}$ lets us rewrite the equation:

$$||\mathbf{X}\omega - \mathbf{y}||^2 - \lambda||\omega||^2$$

Does this look familiar?

Optimisation Problem (1/2)

This leads us to the optimisation problem: Find ω such that the overall loss for all patterns $\mathbf{x}_i$

$$\frac{1}{m}\sum_{i=1}^{m}(\omega \bullet \mathbf{x}_i - y_i)^2 - \lambda||\omega||^2$$

is minimal. Putting all Patterns $\mathbf{x}_i$ into a Matrix $\mathbf{X}$ such that each pattern corresponds to a row in $\mathbf{X}$, and all labels into a vector $\mathbf{y}$ lets us rewrite the equation:

$$||\mathbf{X}\omega - \mathbf{y}||^2 - \lambda||\omega||^2$$

Does this look familiar?
Ridge Regression!

Optimisation Problem (2/2)

$$\omega = \arg\min_{\omega} ||\mathbf{X}\omega - \mathbf{y}||^2 - \lambda||\omega||^2$$

Solution

By deriving w.r.t ω and setting the derivative to zero we can get the solution of this problem:

$$\omega = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I_n})^{-1}\mathbf{X}^T\mathbf{y}$$

where $\mathbf{I_n}$ denotes the $(n \times n)$ Identity matrix. Note that $\mathbf{X}^+ := (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I_n})^{-1}\mathbf{X}^T$ is the Pseudoinverse of $\mathbf{X}$.

Kernelization (1/3)

Room for improvement

This is a really simple approach, and in addition the classifier is a *linear* function on the feature space. Since we know that Kernels are cool, lets try to improve the method by using kernels!

Kernelization (1/3)

Room for improvement

This is a really simple approach, and in addition the classifier is a *linear* function on the feature space. Since we know that Kernels are cool, lets try to improve the method by using kernels!

Feature Transform

Idea: Take our patterns, transform them to some other space, and do the training and classification there. Transform will be done by a function

$$\phi(\mathbf{x}) : \mathcal{X}^n \rightarrow \Phi^N$$

Define kernel function $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \bullet \phi(\mathbf{y})$

Kernelization (2/3)

Restating our problem

Now our decision function looks like this:

$$f(\mathbf{x}) = \alpha \bullet \phi(\mathbf{x})$$

where α is a new weight vector in Φ-space of dim. $(N \times 1)$

Kernelization (2/3)

Restating our problem

Now our decision function looks like this:

$$f(\mathbf{x}) = \alpha \bullet \phi(\mathbf{x})$$

where α is a new weight vector in Φ-space of dim. $(N \times 1)$

Inserting $\phi(\mathbf{x})$ into the solution

All we did was change the representation of $\mathbf{x}$, so the optimisation step for finding α is still the same:

$$\alpha = (\phi(\mathbf{X})^T \phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1} \phi(\mathbf{X})^T \mathbf{y}$$

Note that $\phi(\mathbf{X})$ is the transform of the whole data Matrix by ϕ

Kernelization (3/3)

There is a theorem that tells us that for the pseudoinverse it holds

$$(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T = \phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}$$

by applying this equality to the solution of the regression problem we can find that:

$$f(\mathbf{x}) \quad = \phi(\mathbf{x}) \bullet \alpha$$

Kernelization (3/3)

There is a theorem that tells us that for the pseudoinverse it holds

$$(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T = \phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}$$

by applying this equality to the solution of the regression problem we can find that:

$$\begin{aligned} f(\mathbf{x}) &= \phi(\mathbf{x}) \bullet \alpha \\ &= \phi(\mathbf{x})^T(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T\mathbf{y} \end{aligned}$$

Kernelization (3/3)

There is a theorem that tells us that for the pseudoinverse it holds

$$(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T = \phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}$$

by applying this equality to the solution of the regression problem we can find that:

$$\begin{aligned} f(\mathbf{x}) &= \phi(\mathbf{x}) \bullet \alpha \\ &= \phi(\mathbf{x})^T(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T\mathbf{y} \\ &= \phi(\mathbf{x})^T\phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}\mathbf{y} \end{aligned}$$

Kernelization (3/3)

There is a theorem that tells us that for the pseudoinverse it holds

$$(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T = \phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}$$

by applying this equality to the solution of the regression problem we can find that:

$$\begin{aligned} f(\mathbf{x}) &= \phi(\mathbf{x}) \bullet \alpha \\ &= \phi(\mathbf{x})^T(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T\mathbf{y} \\ &= \phi(\mathbf{x})^T\phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}\mathbf{y} \\ &= \phi(\mathbf{x})^T\phi(\mathbf{X})^T \underbrace{(\mathbf{K} + \lambda \mathbf{I_m})^{-1}\mathbf{y}}_{\mathbf{w}} \end{aligned}$$

Kernelization (3/3)

There is a theorem that tells us that for the pseudoinverse it holds

$$(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T = \phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}$$

by applying this equality to the solution of the regression problem we can find that:

$$\begin{aligned} f(\mathbf{x}) &= \phi(\mathbf{x}) \bullet \alpha \\ &= \phi(\mathbf{x})^T(\phi(\mathbf{X})^T\phi(\mathbf{X}) + \lambda \mathbf{I_n})^{-1}\phi(\mathbf{X})^T\mathbf{y} \\ &= \phi(\mathbf{x})^T\phi(\mathbf{X})^T(\phi(\mathbf{X})\phi(\mathbf{X})^T + \lambda \mathbf{I_m})^{-1}\mathbf{y} \\ &= \phi(\mathbf{x})^T\phi(\mathbf{X})^T \underbrace{(\mathbf{K} + \lambda \mathbf{I_m})^{-1}\mathbf{y}}_{\mathbf{w}} \\ &= \sum_{i=1}^{m} w_i k(\mathbf{x}, \mathbf{x}_i) \end{aligned}$$

All Transforms are now contained in Kernel functions!

Summary

So we have seen that we can rewrite the decision function in the input space as

$$f(\mathbf{x}) = \sum_{i=1}^{m} w_i k(\mathbf{x}, \mathbf{x}_i)$$

and we can find the weights by solving the linear equation

$$\mathbf{w} = (\mathbf{K} + \lambda \mathbf{I_m})^{-1} \mathbf{y}$$

$\mathbf{K}$ is the Kernel Matrix defined by $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ for all training patterns $\mathbf{x}_i$ and $\mathbf{x}_j$. Note that $\mathbf{w}$ is of dimension m.

Ensemble Classifiers (1/3)

Idea

Instead of creating *one single* highly sophisticated classifier, let's take *many simple* classifiers (weak learners), and do majority voting.

Formalization

We take a number m of simple classifiers $f_i(\mathbf{x})$, and create a decision function

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} f_i(\mathbf{x})\right)$$

Ensemble Classifiers (2/3)

Bias - Variance Tradeoff

We have seen in the last lecture that the expected classification error can be decomposed into

$$E_D\left[f(\mathbf{x}, D) - y\right]^2 = \underbrace{\left[E_D f(\mathbf{x}, D) - y\right]^2}_{Bias^2} + \underbrace{E_D\left[f(\mathbf{x}, D) - E_D f(\mathbf{x}, D)\right]^2}_{Variance}$$

where E_D denotes the expectation over the training set D.

Ensemble Classifiers (2/3)

Bias - Variance Tradeoff

We have seen in the last lecture that the expected classification error can be decomposed into

$$E_D\left[f(\mathbf{x}, D) - y\right]^2 = \underbrace{\left[E_D f(\mathbf{x}, D) - y\right]^2}_{Bias^2} + \underbrace{E_D\left[f(\mathbf{x}, D) - E_D f(\mathbf{x}, D)\right]^2}_{Variance}$$

where E_D denotes the expectation over the training set D.

Variance Reduction

We note that our ensemble classifier in fact *averages* over m weak learners.

$\Rightarrow$ the resulting classifier will have less variance

Ensemble Classifiers (2/3)

Bias - Variance Tradeoff

We have seen in the last lecture that the expected classification error can be decomposed into

$$E_D\left[f(\mathbf{x}, D) - y\right]^2 = \underbrace{\left[E_D f(\mathbf{x}, D) - y\right]^2}_{Bias^2} + \underbrace{E_D\left[f(\mathbf{x}, D) - E_D f(\mathbf{x}, D)\right]^2}_{Variance}$$

where E_D denotes the expectation over the training set D.

Variance Reduction

We note that our ensemble classifier in fact *averages* over m weak learners.

$\Rightarrow$ the resulting classifier will have less variance

$\Rightarrow$ ensemble classifiers reduce variance!

Ensemble Classifiers (3/3)

Bias

We want the expected classification error as low as possible.
Variance is already taken care of by the averaging process.
$\Rightarrow$ We have to look for weak learners with low bias

Ensemble Classifiers (3/3)

Bias

We want the expected classification error as low as possible.
Variance is already taken care of by the averaging process.
$\Rightarrow$ We have to look for weak learners with low bias

How to reduce bias

The bias-variance tradeoff also holds for single classifiers. So we can reduce bias by increasing variance.
$\Rightarrow$ Overfit the individual classifiers!

Feature Selection (1/2)

Contest Data contains large number of features. Many non-informative and even garbage features. Use feature selection technique to make classification faster and better.
The authors used random forest to calculate a relevance index for the features.

Random Forest

RF creates a number N_T of decision trees. At each node of the decision tree a random subset of features is selected, and out of this subset the variable that reduces entropy most is selected as splitter.

Variable Selection with RF

Relevance Index

At a given tree T for each variable we sum up entropy reduction at each node where the data was split according to this variable.

$$M(x_i, T) = \sum_{t \in T | x_i \text{ is splitter variable at node } t} \Delta I(x_i, t)$$

Variable relevance can then be defined as the average entropy reduction over all trees T.

$$M(x_i) = \frac{1}{N_T} \sum_{k=1}^{N_T} M(x_i, T_k)$$

Experiments (1/3))

Feature Selection

The authors used the relevance index created by RF to look for cutoff points, i.e. sudden drops in the relevance of variables, to get candidates for split into relevant/non relevant variables. The optimal cutoff point was selected by cross validation (CV).

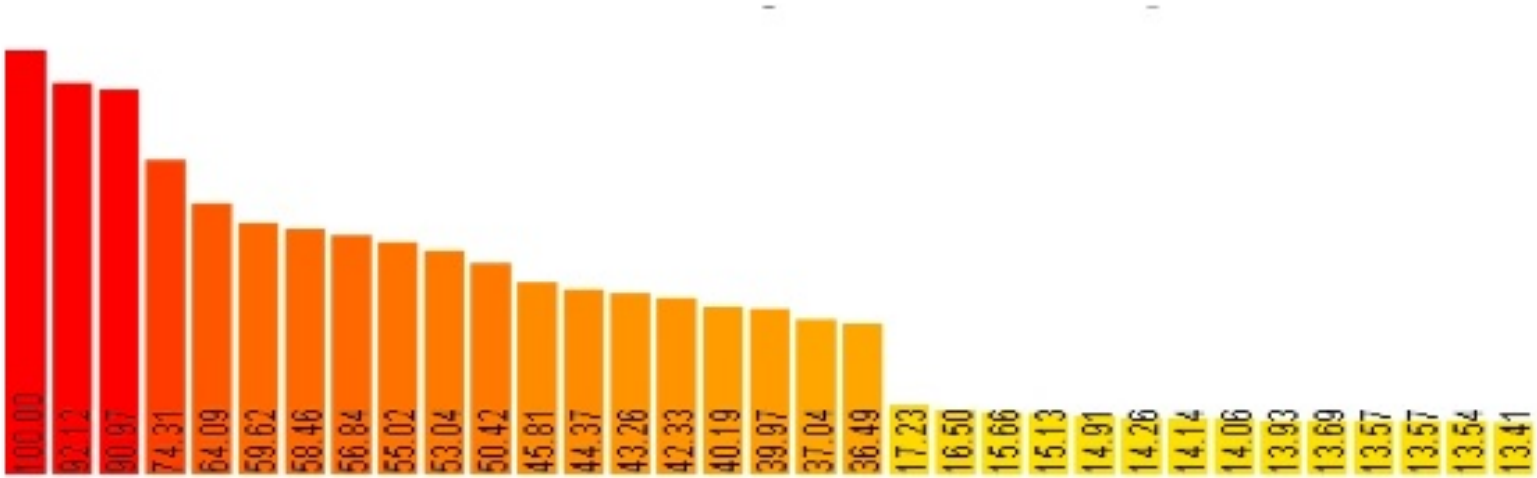


Figure: Relevance of top 33 features of Madelon

Experiments (2/3)

Preprocessing and Feature Selection

With some datasets also standardization and variable weighting improved performance.

Data set	Original variables	Selected variables	Selection method	Stand-ardize?	Weight-ing?
MADELON	500	19	RF	yes	no
DEXTER	20000	500	MMI	yes	by MI
ARCENE	10000	10000	none	no	no
GISETTE	5000	307	RF	no	no
DOROTHEA	100000	284	RF	no	no

Figure: Optimal Preprocessing settings found by crossvalidation

Experiments (3/3))

ensemble vs. single classifier

The authors were also interested in the difference between an ensemble of regularized least squares classifiers and one single RLSC. For that purpose they trained both versions and compared errorrates.

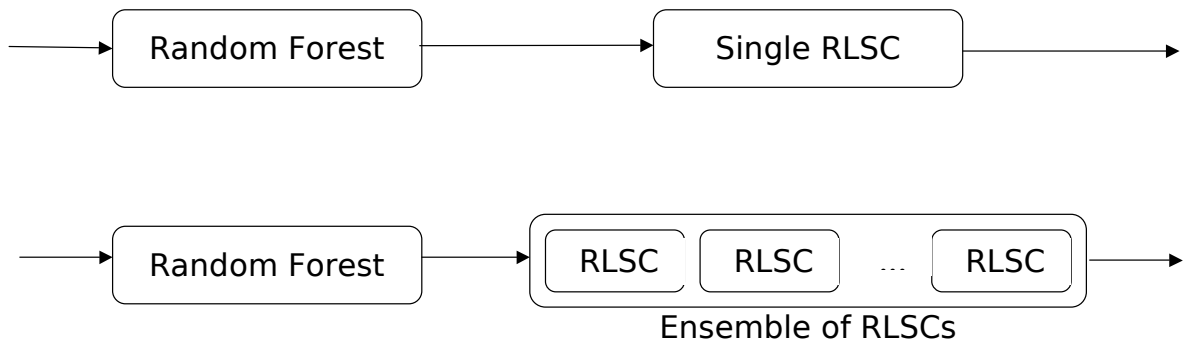


Figure: Experimental Setup

Particular Settings

- Gaussian Kernels with spherical covariance matrices were chosen, the optimal kernel width σ^2 was found by CV.
- The regularisation parameter was also found by CV for the single RLSC, for the ensemble it was kept very small in order to allow overfitting.
- The ensemble size didn't really make great differences as long as it was "reasonably large" (around 200)
- The individual RLSC in the ensemble were trained with a random subset of the training data, chosen *without* replacement, since duplicates only result in a duplicate kernel function in the solution.

CV error for ten-fold CV

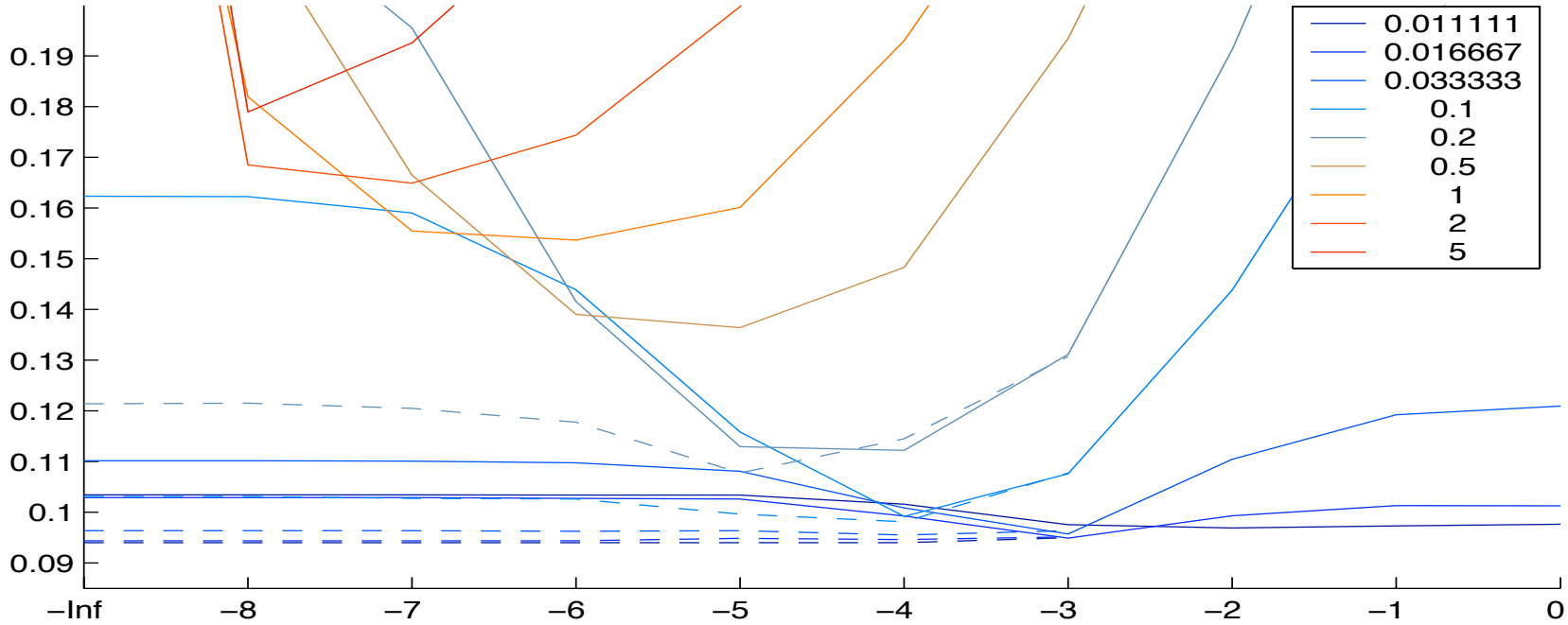
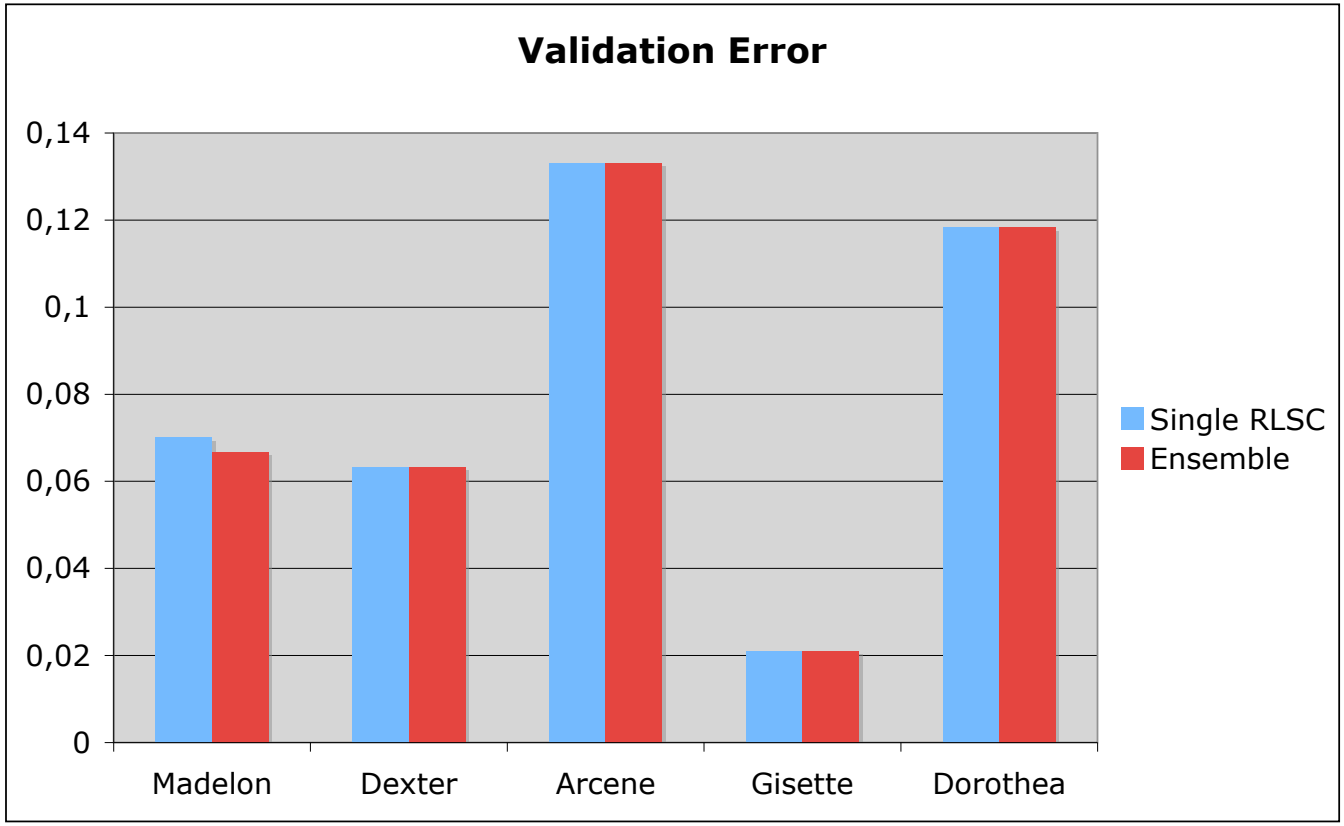


Figure: Experimental Setup

The horizontal axis corresponds to $\log_{10}(\lambda)$, the vertical axis is the CV-error. Each curve corresponds to a particular kernel width. Dashed curves denote the ensemble, solid curves the single RLSC.

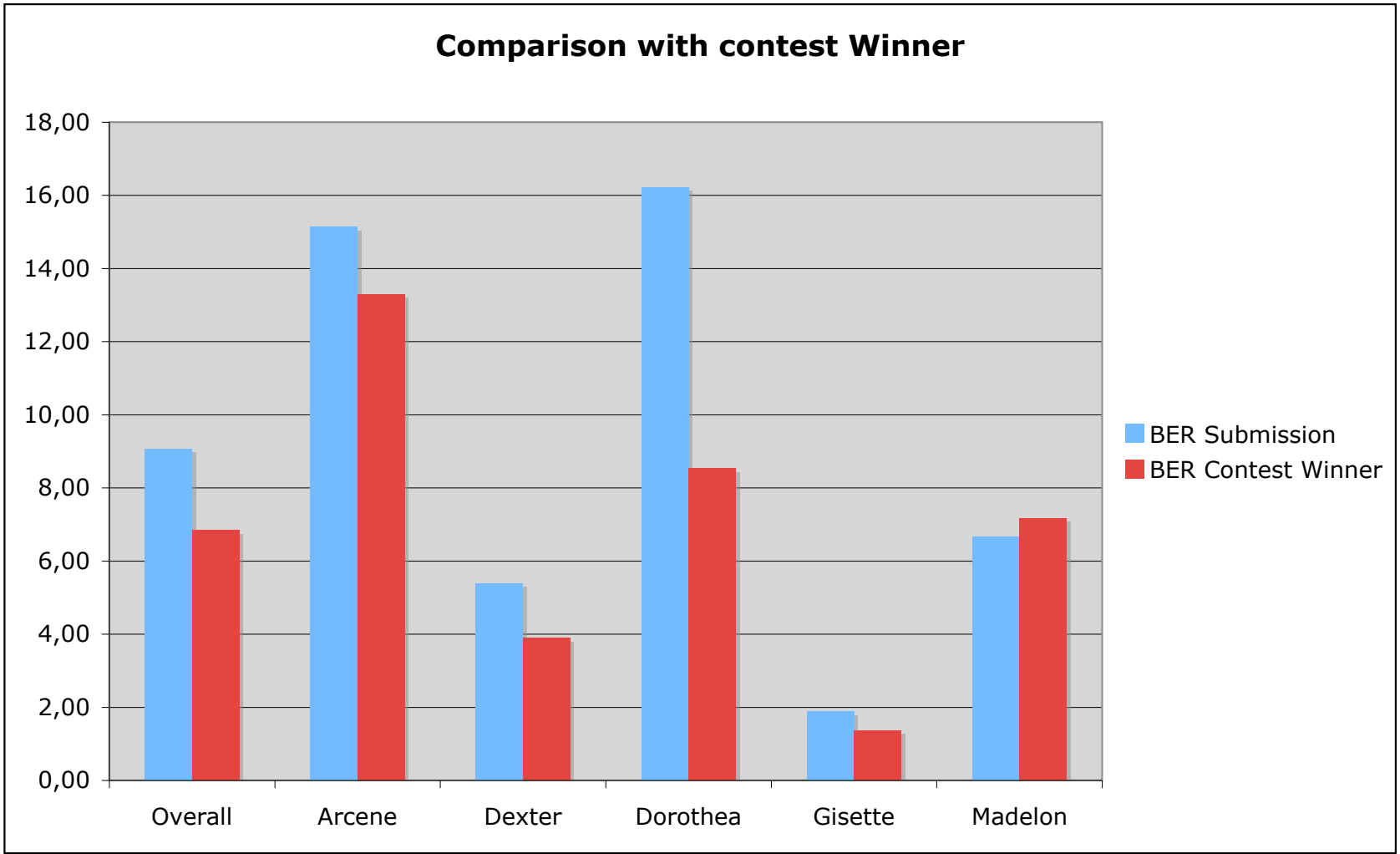
Ensemble vs. Single RLSC



It can be seen that there are no significant differences between the ensemble and the single classifier on the validation set.

NIPS 2003 Challenge: Results

For the NIPS 2003 Challenge the authors submitted the ensemble classifier, because the ensemble is more robust (less sensitive to parameter choices).



Conclusion

- The experiments show empirically that aggregation of multiple (non regularized) RLSC classifiers delivers the same results as *one* regularized RLSC
- So bagging unstable classifiers (classifiers with high variance) makes them stable.
- Advantage: ensembles seem to be more robust w.r.t. hyperparameter selection.
- Very simple but powerful method
- Can be easily parallelized

Remarks

The Quest for low Bias

Weak learners used for ensemble classifiers should have low bias as pointed out before. However the model that the authors propose implicitly introduces bias at two places

- Choice of the loss function: Quadratic Loss penalizes outliers more than points close to the decision boundary.
- Regularisation

Remarks

The Quest for low Bias

Weak learners used for ensemble classifiers should have low bias as pointed out before. However the model that the authors propose implicitly introduces bias at two places

- Choice of the loss function: Quadratic Loss penalizes outliers more than points close to the decision boundary.
- Regularisation

Feature Selection vs. good Classifier

The authors combine Random Forest and an ensemble of RLSC. Results are very good, but it is not clear if this is due to the classifier or due to the feature selection process.