Binary/Ternary Extreme Learning Machines

Mark van Heeswijk, Yoan Miche

October 3, 2013

Outline

Motivation

Binary / Ternary ELM

Experiments

Conclusion

Outline

Standard ELM

Given a training set $(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}$, an activation function $f : \mathbb{R} \mapsto \mathbb{R}$ and M the number of hidden nodes:

1: - Randomly assign input weights $\mathbf{w}_i$ and biases b_i, $i \in [1, M]$;
2: - Calculate the hidden layer output matrix $\mathbf{H}$;
3: - Calculate output weights matrix $\beta = \mathbf{H}^{\dagger}\mathbf{Y}$.

ELM Theory vs Practice

- In theory, ELM is universal approximator
- In practice, limited number of samples; risk of overfitting
- Therefore:
  - the functional approximation should use as limited number of neurons as possible
  - the hidden layer should extract and retain as much information as possible from the input samples

which neurons work well together to extract as much useful information as possible?

Better Weights

- random layer weights and biases drawn from e.g. uniform / normal distribution with certain range / variance
- typical transfer function $f\left(\langle \mathbf{w_i}, \mathbf{x} \rangle + b_i\right)$
- from $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta$, it can be seen that the typical activation of f depends on:
  - expected length of $\mathbf{w_i}$
  - expected length of $\mathbf{x}$
  - angles θ between the weights and the samples

Better Weights: Orthogonality?

Idea 1:

- improve the diversity of the weights by taking weights that are mutually orthogonal (e.g. the M d-dimensional basis vectors, randomly rotated in the d-dimensional space)
- however, does not give significantly better accuracy :(
- apparently, for the tested cases, random weight scheme of ELM already covers the possible weight space pretty well

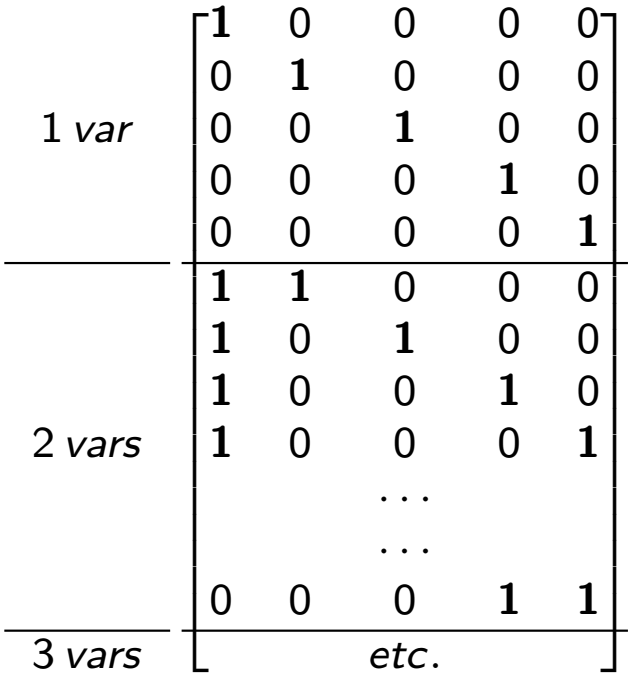
Better Weights: Sparsity!

Idea 2:

- improve the diversity of the weights by having each of them work in a different subspace (e.g. each weight vector has different subset of variables as input)
- spoiler: significantly improves accuracy, at no extra computational cost :)
- experiments suggest this is due to the weight scheme enabling implicit variable selection

Outline

Binary Weight Scheme



until enough neurons:

- add $\mathbf{w} \in \{0,1\}^d$ with 1 var $(\# = 2^1 \times \binom{d}{1})$
- add $\mathbf{w} \in \{0,1\}^d$ with 2 vars $(\# = 2^2 \times \binom{d}{2})$
- add $\mathbf{w} \in \{0,1\}^d$ with 3 vars $(\# = 2^3 \times \binom{d}{3})$
- ...

For each subspace, weights are added in random order to avoid bias toward particular variables

Ternary Weight Scheme

1 *var*	$+\mathbf{1}\ 0\ 0\ 0$ $-\mathbf{1}\ 0\ 0\ 0$ $0\ +\mathbf{1}\ 0\ 0$ $0\ -\mathbf{1}\ 0\ 0$
2 *vars*	$+\mathbf{1}\ +\mathbf{1}\ 0\ 0$ $+\mathbf{1}\ -\mathbf{1}\ 0\ 0$ $-\mathbf{1}\ +\mathbf{1}\ 0\ 0$ $-\mathbf{1}\ -\mathbf{1}\ 0\ 0$
3 *vars*	$0\ 0\ -\mathbf{1}\ -\mathbf{1}$

until enough neurons:

- add $\mathbf{w} \in \{-1, 0, 1\}^d$ with 1 var $(3^1 \times \binom{d}{1})$
- add $\mathbf{w} \in \{-1, 0, 1\}^d$ with 2 vars $(3^2 \times \binom{d}{2})$
- add $\mathbf{w} \in \{-1, 0, 1\}^d$ with 3 vars $(3^3 \times \binom{d}{3})$
- ...

For each subspace, weights are added in random order to avoid bias toward particular variables

Some Notes

Weight considerations:

- weight range determines typical activation of the transfer function (remember $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta,$)
- therefore, normalize or tune somehow using e.g. Batch Intrinsic Plasticity Pretraining
- any performance difference between weight schemes will therefore come from the different directions of the weights

Linear vs non-linear:

- since sigmoid neurons operate in nonlinear regime, add d linear neurons for the ELM to work better on (almost) linear problems

Avoiding overfitting:

- use efficient L2 regularization

Some Notes

Weight considerations:

- weight range determines typical activation of the transfer function (remember $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta,$)
- therefore, normalize or tune somehow using e.g. Batch Intrinsic Plasticity Pretraining
- any performance difference between weight schemes will therefore come from the different directions of the weights

Linear vs non-linear:

- since sigmoid neurons operate in nonlinear regime, add d linear neurons for the ELM to work better on (almost) linear problems

Avoiding overfitting:

- use efficient L2 regularization

Some Notes

Weight considerations:

- weight range determines typical activation of the transfer function (remember $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta,$)
- therefore, normalize or tune somehow using e.g. Batch Intrinsic Plasticity Pretraining
- any performance difference between weight schemes will therefore come from the different directions of the weights

Linear vs non-linear:

- since sigmoid neurons operate in nonlinear regime, add d linear neurons for the ELM to work better on (almost) linear problems

Avoiding overfitting:

- use efficient L2 regularization

Some Notes

Weight considerations:

- weight range determines typical activation of the transfer function (remember $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta,$)
- therefore, normalize or tune somehow using e.g. Batch Intrinsic Plasticity Pretraining
- any performance difference between weight schemes will therefore come from the different directions of the weights

Linear vs non-linear:

- since sigmoid neurons operate in nonlinear regime, add d linear neurons for the ELM to work better on (almost) linear problems

Avoiding overfitting:

- use efficient L2 regularization

Some Notes

Weight considerations:

- weight range determines typical activation of the transfer function (remember $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta,$)
- therefore, normalize or tune somehow using e.g. Batch Intrinsic Plasticity Pretraining
- any performance difference between weight schemes will therefore come from the different directions of the weights

Linear vs non-linear:

- since sigmoid neurons operate in nonlinear regime, add d linear neurons for the ELM to work better on (almost) linear problems

Avoiding overfitting:

- use efficient L2 regularization

Some Notes

Weight considerations:

- weight range determines typical activation of the transfer function (remember $\langle \mathbf{w_i}, \mathbf{x} \rangle = |\mathbf{w_i}||\mathbf{x}| \cos\theta,$)
- therefore, normalize or tune somehow using e.g. Batch Intrinsic Plasticity Pretraining
- any performance difference between weight schemes will therefore come from the different directions of the weights

Linear vs non-linear:

- since sigmoid neurons operate in nonlinear regime, add d linear neurons for the ELM to work better on (almost) linear problems

Avoiding overfitting:

- use efficient L2 regularization

Outline

Experimental Settings

Data	Abbreviation	number of variables	# training	# test
Abalone	**Ab**	**8**	**2000**	**2177**
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	**Co**	**12**	**4000**	**4192**

- BIP(CV)-TR-ELM vs BIP(CV)-TR-2-ELM vs BIP(CV)-TR-3-ELM
- Experiment 1: relative performance
- Experiment 2: robustness against irrelevant vars
- Experiment 3: implicit variable selection
- (all results are averaged over 100 repetitions, each with randomly drawn training/test set)

Experimental Settings

Data	Abbreviation	number of variables	# training	# test
Abalone	**Ab**	**8**	**2000**	**2177**
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	**Co**	**12**	**4000**	**4192**

- BIP(CV)-TR-ELM vs BIP(CV)-TR-2-ELM vs BIP(CV)-TR-3-ELM
- Experiment 1: relative performance
- Experiment 2: robustness against irrelevant vars
- Experiment 3: implicit variable selection
- (all results are averaged over 100 repetitions, each with randomly drawn training/test set)

Experimental Settings

Data	Abbreviation	number of variables	# training	# test
Abalone	**Ab**	**8**	**2000**	**2177**
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	**Co**	**12**	**4000**	**4192**

- BIP(CV)-TR-ELM vs BIP(CV)-TR-2-ELM vs BIP(CV)-TR-3-ELM
- Experiment 1: relative performance
- Experiment 2: robustness against irrelevant vars
- Experiment 3: implicit variable selection
- (all results are averaged over 100 repetitions, each with randomly drawn training/test set)

Experimental Settings

Data	Abbreviation	number of variables	# training	# test
Abalone	**Ab**	**8**	**2000**	**2177**
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	**Co**	**12**	**4000**	**4192**

- BIP(CV)-TR-ELM vs BIP(CV)-TR-2-ELM vs BIP(CV)-TR-3-ELM
- Experiment 1: relative performance
- Experiment 2: robustness against irrelevant vars
- Experiment 3: implicit variable selection
- (all results are averaged over 100 repetitions, each with randomly drawn training/test set)

Experimental Settings

Data	Abbreviation	number of variables	# training	# test
Abalone	**Ab**	**8**	**2000**	**2177**
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	**Co**	**12**	**4000**	**4192**

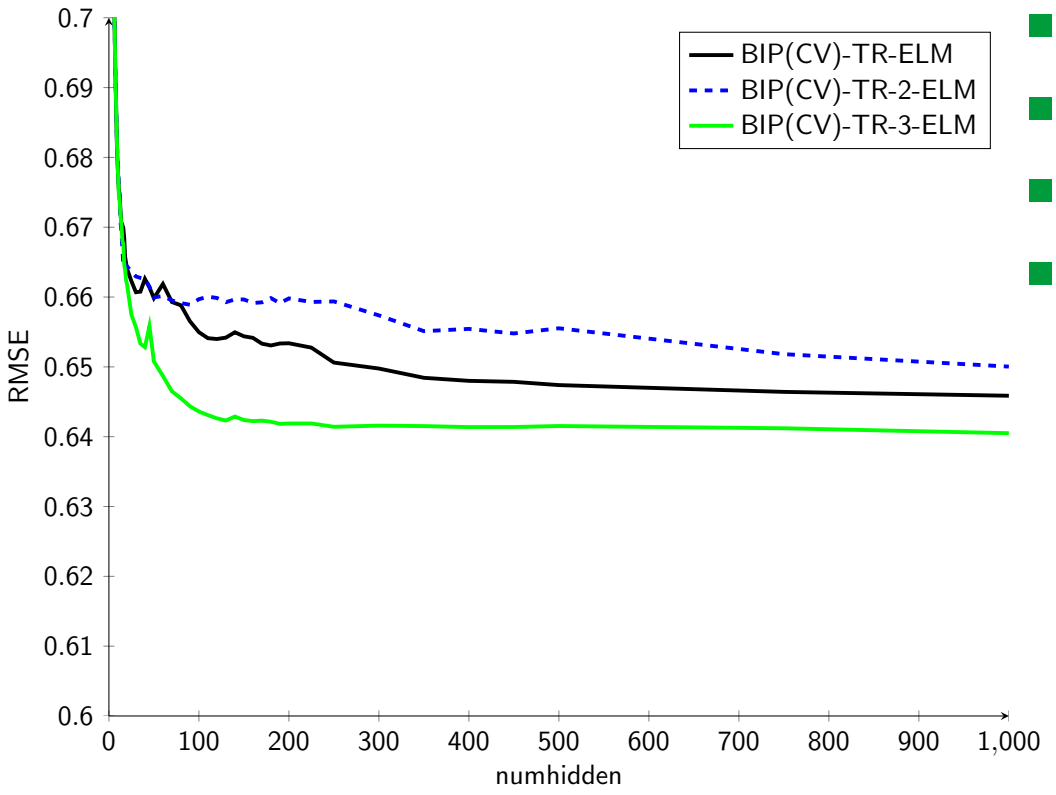
- BIP(CV)-TR-ELM vs BIP(CV)-TR-2-ELM vs BIP(CV)-TR-3-ELM
- Experiment 1: relative performance
- Experiment 2: robustness against irrelevant vars
- Experiment 3: implicit variable selection
- (all results are averaged over 100 repetitions, each with randomly drawn training/test set)

Experimental Settings

Data	Abbreviation	number of variables	# training	# test
Abalone	**Ab**	**8**	**2000**	**2177**
CaliforniaHousing	Ca	8	8000	12640
CensusHouse8L	Ce	8	10000	12784
DeltaElevators	De	6	4000	5517
ComputerActivity	**Co**	**12**	**4000**	**4192**

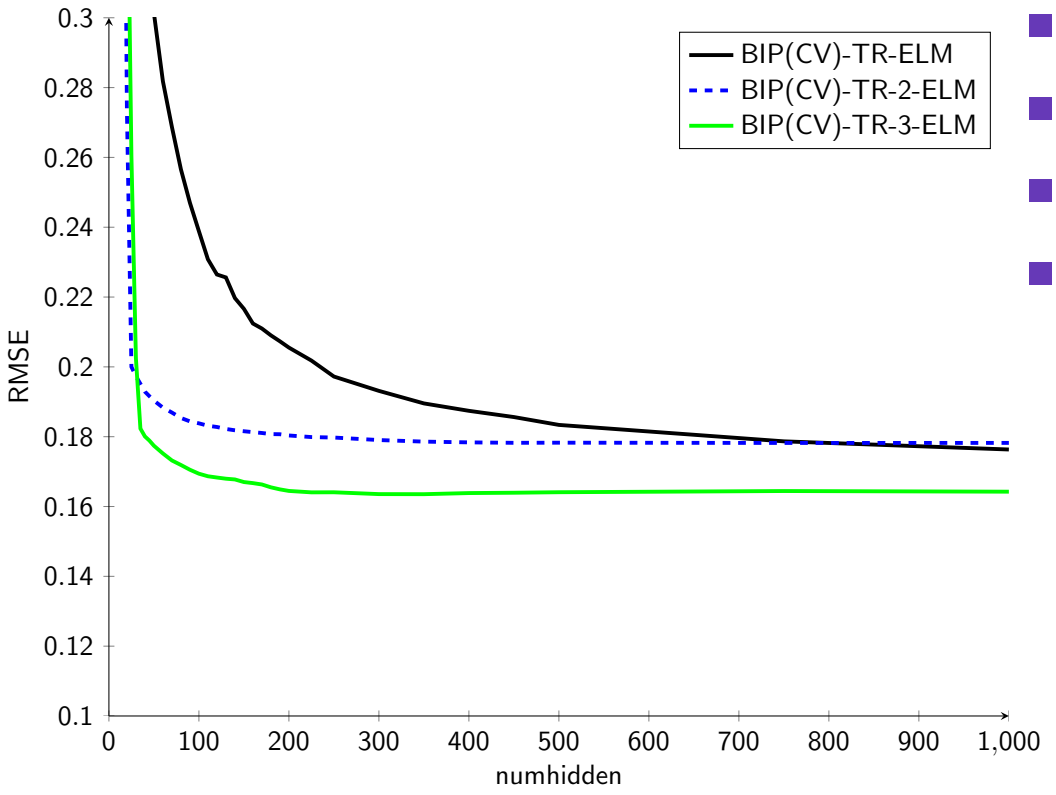
- BIP(CV)-TR-ELM vs BIP(CV)-TR-2-ELM vs BIP(CV)-TR-3-ELM
- Experiment 1: relative performance
- Experiment 2: robustness against irrelevant vars
- Experiment 3: implicit variable selection
- (all results are averaged over 100 repetitions, each with randomly drawn training/test set)

Exp 1: numhidden vs. RMSE (Abalone)



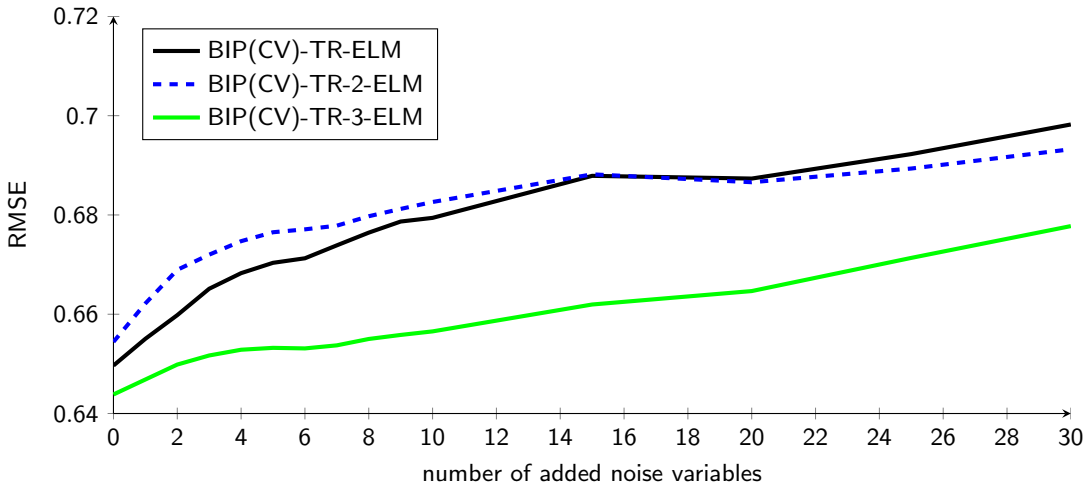
- averages over 100 runs
- gaussian $<$ binary
- ternary $<$ gaussian
- better RMSE with much less neurons

Exp 1: numhidden vs. RMSE (CpuActivity)



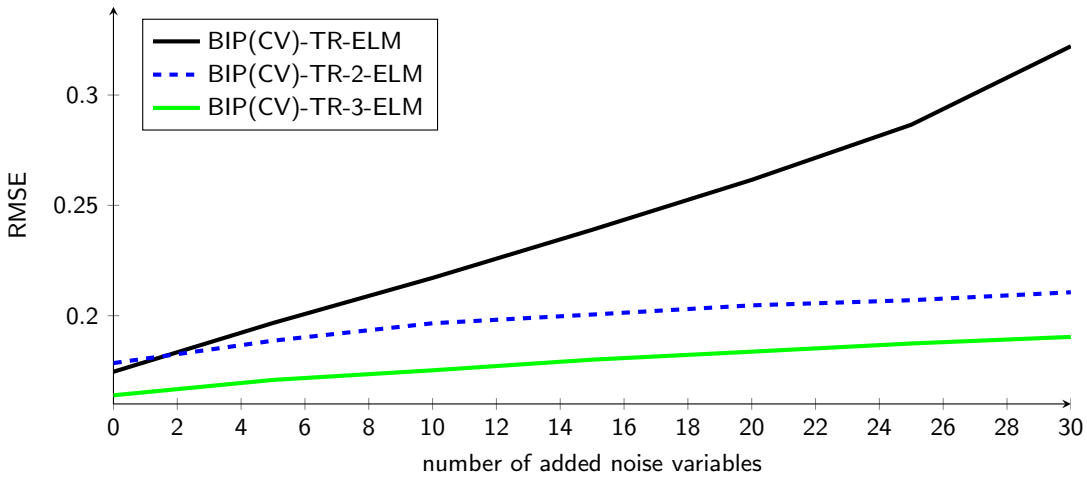
- averages over 100 runs
- binary $<$ gaussian
- ternary $<$ gaussian
- better RMSE with much less neurons

Exp 2: Robustness against irrelevant variables (Abalone)



- 1000 neurons
- binary weight scheme gives similar RMSE
- ternary weight scheme makes ELM more robust against irrelevant vars

Exp 2: Robustness against irrelevant variables (CpuActivity)



- 1000 neurons
- binary and ternary weight scheme makes ELM more robust against irrelevant vars

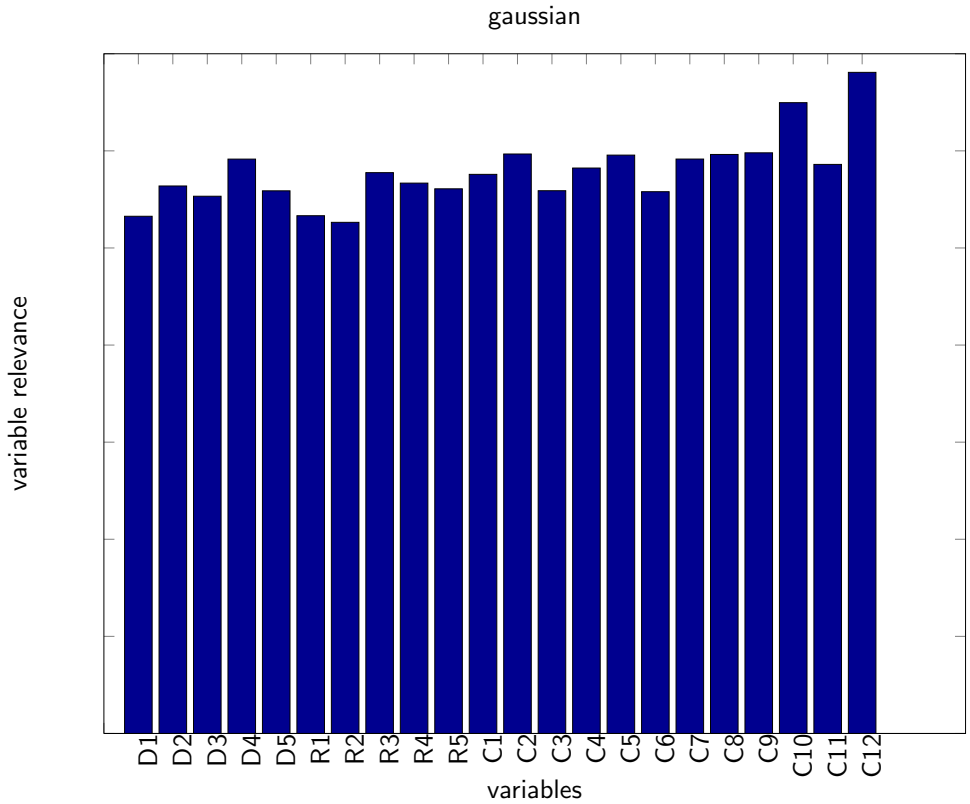
Exp 2: Robustness against irrelevant variables

	Ab			Co		
	gaussian	binary	ternary	gaussian	binary	ternary
RMSE with original variables	0.6497	0.6544	0.6438	0.1746	0.1785	0.1639
RMSE with 30 added irr. vars	0.6982	0.6932	0.6788	0.3221	0.2106	0.1904
RMSE loss	0.0486	**0.0388**	**0.0339**	0.1475	**0.0321**	**0.0265**

Table: Average RMSE loss of ELMs with 1000 hidden neurons, trained on the original data, and the data with 30 added irrelevant variables

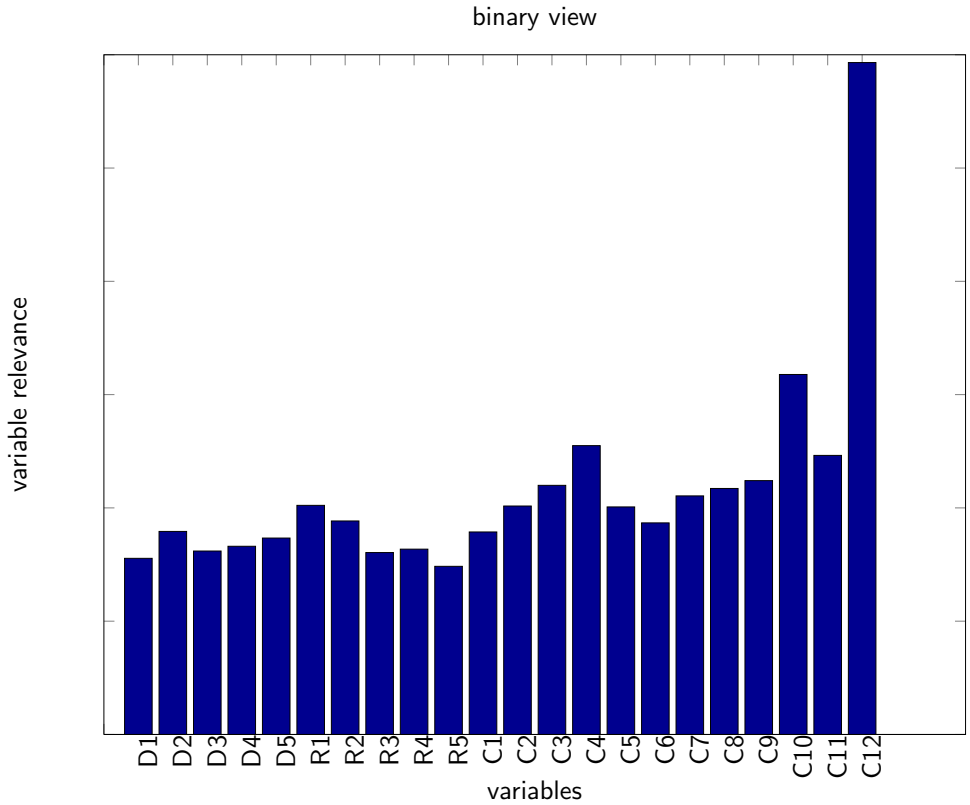
Exp 3: Implicit Variable Selection (CpuAct)

- relevance of each input variable quantified as $\sum_{i=1}^{M} |\beta_i \times \mathbf{w}_i|$



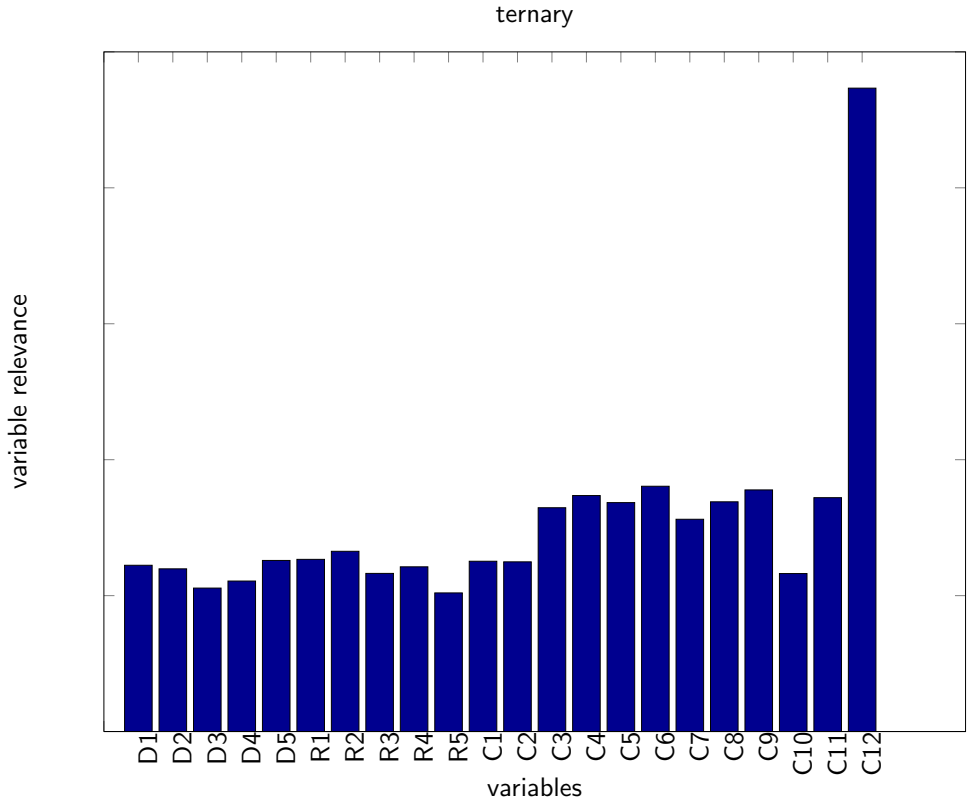
Exp 3: Implicit Variable Selection (CpuAct)

- relevance of each input variable quantified as $\sum_{i=1}^{M} |\beta_i \times \mathbf{w}_i|$



Exp 3: Implicit Variable Selection (CpuAct)

- relevance of each input variable quantified as $\sum_{i=1}^{M} |\beta_i \times \mathbf{w}_i|$



Outline

Conclusions

We propose simple change to weight scheme and introduce robust ELM variants:

- BIP(CV)-TR-ELM
- BIP(CV)-TR-2-ELM
- BIP(CV)-TR-3-ELM

Our experiments suggest that

1. ternary weight scheme generally better than gaussian weights
2. ternary weight scheme robust against irrelevant variables
3. binary/ternary weight scheme allows ELM to perform implicit variable selection

The added robustness and increased accuracy comes for free!

Questions?

Batch Intrinsic Plasticity

- suppose $(\mathbf{x}_1, ..., \mathbf{x}_N) \in \mathbb{R}^{N \times d}$, and output of neuron i is $h_i = f(a_i \mathbf{w}_i \cdot \mathbf{x}_k + b_i)$, where f is an invertible transfer function
- for each neuron i
  - from exponential distribution with mean μ_{exp}, draw targets $\mathbf{t} = (t_1, t_2, \ldots, t_N)$ and sort such that $t_1 < t_2 < \ldots < t_N$
  - compute all presynaptic inputs $\mathbf{s}_k = \mathbf{w}_i \cdot \mathbf{x}_k$, and sort such that $s_1 < s_2 < \ldots < s_N$
  - now, find a_i and b_i such that

$$\begin{pmatrix} s_1 & 1 \\ \vdots & 1 \\ s_N & 1 \end{pmatrix} \begin{pmatrix} a_i \\ b_i \end{pmatrix} = \begin{pmatrix} f^{-1}(t_1) \\ \vdots \\ f^{-1}(t_N) \end{pmatrix}$$

Fast leave-one-out cross-validation

The leave-one-out (LOO) error can be computed using the PRESS statistics:

$$E_{loo} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{y_i - \hat{y}_i}{1 - hat_{ii}} \right)^2$$

where hat_{ii} is the i^{th} value on the diagonal of the HAT-matrix, which can be quickly computed, given $\mathbf{H}^{\dagger}$:

$$\begin{aligned} \hat{\mathbf{Y}} &= \mathbf{H}\beta = \mathbf{H}\mathbf{H}^{\dagger}\mathbf{Y} \\ &= \mathrm{HAT} \cdot \mathbf{Y} \end{aligned}$$

Fast leave-one-out cross-validation

Using the SVD decomposition of $\mathbf{H} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, it is possible to obtain all needed information for computing the PRESS statistic without recomputing the pseudo-inverse for every λ:

$$
\begin{aligned}
\hat{\mathbf{Y}} &= \mathbf{H}\beta \\
&= \mathbf{H}(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^T\mathbf{Y} \\
&= \mathbf{H}\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}\mathbf{U}^T\mathbf{Y} \\
&= \mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{V}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}\mathbf{U}^T\mathbf{Y} \\
&= \mathbf{U}\mathbf{D}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}\mathbf{U}^T\mathbf{Y} \\
&= \mathrm{HAT} \cdot \mathbf{Y}
\end{aligned}
$$

Fast leave-one-out cross-validation

where $\mathbf{D}(\mathbf{D}^2 + \lambda\mathbf{I})^{-1}\mathbf{D}$ is a diagonal matrix with $\frac{d_{ii}^2}{d_{ii}^2+\lambda}$ as the i^{th} diagonal entry. Now:

$$
\begin{aligned}
\mathrm{MSE}^{\text{TR-PRESS}} &= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{1 - hat_{ii}}\right)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{1 - \mathbf{h}_{i\cdot}(\mathbf{H}^T\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{h}_{i\cdot}^T}\right)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{y}_i}{1 - \mathbf{u}_{i\cdot}\left(\frac{d_{ii}^2}{d_{ii}^2+\lambda}\right)\mathbf{u}_{i\cdot}^T}\right)^2
\end{aligned}
$$