Neural Network Interest Group

Título/*Title*:
Resilient Methods for Shannon Entropy and Z-EDM

Autor(es)/*Author(s)*:
Luís M. Silva

Relatório Técnico/*Technical Report* No. 5 /2007

FEUP/INEB, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Título/*Title*:

Resilient Methods for Shannon Entropy and Z-EDM

Autor(es)/*Author(s)*:

Luís M. Silva

Relatório Técnico/*Technical Report* No. 5 /2007

Publicado por/*Published by*: NNIG. http://paginas.fe.up.pt/~nnig/

Contents

Chapter 1

Resilient methods for Shannon entropy

1.1 Resilient methods

In order to speedup the convergence of the proposed algorithm we have selected 3 algorithms from the literature. We present a brief description of each

- Normal
  The first one uses the same strategy as in [1], where a variable learning rate is used to control the descent (in this case the ascent) through the performance surface. In the presence of a decrease of the value of f at the origin from an epoch to another, the network is set to the previous state and the learning rate is decreased, otherwise, η is increased to speedup convergence.

- Rprop
  This method, proposed by Riedmiller and Braun [2, 3], is considered among the best performing first-order learning methods for neural networks. It controls every single weight connection by using individual stepsizes. These stepsizes are updated by only inspecting successive steps of the derivative's signal. Independence from the magnitude of the derivative and equal learning on the entire network are among the benefits of the method cite.

- iRprop
  iRprop stands for Improved Rprop and was proposed by Igel and Husken [4, 5]. In fact, iRprop$^+$ is the improved version used here (see cite for more approaches). The difference to the original Rprop algorithm relies on the combination of individual information about the error surface (sign of the derivative) with more global information (the value of the error itself) to provide a weight-backtracking scheme.

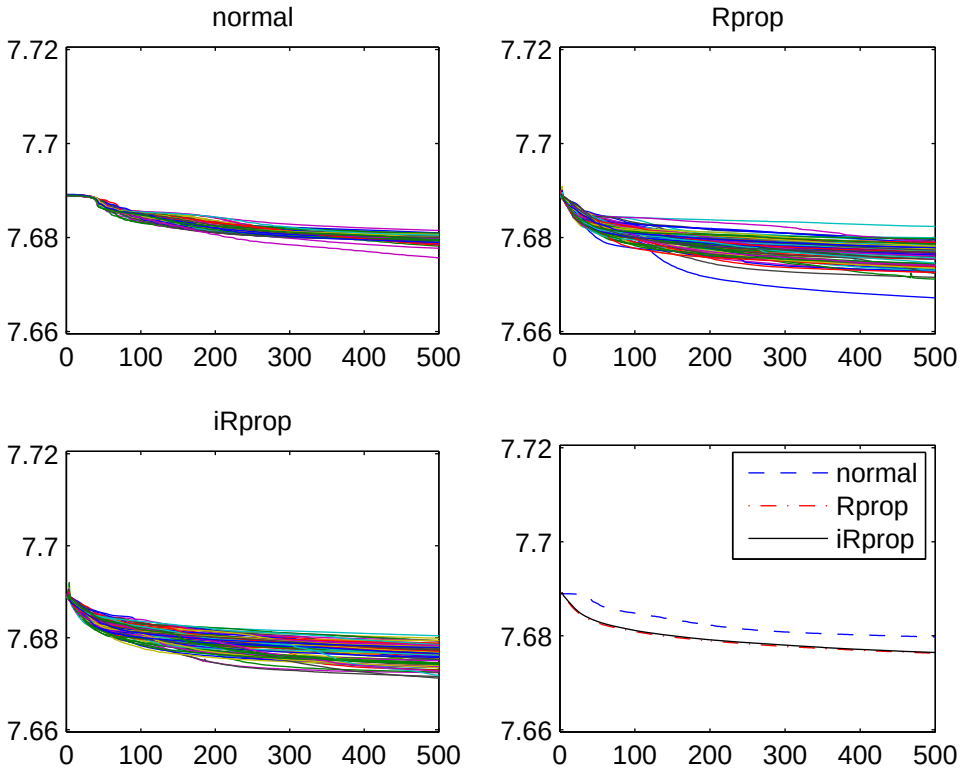
1.2 Experiments and Results

We performed two types of experiments and used 4 well known datasets: Pb12 from [6] and Iris, Wdbc and Pima from the UCI repository [7]. To have a fair comparison, a total of 100 trials of each experiment was made and the initial weights were always the same (for each trial) for each algorithm.

1.2.1 Evaluating convergence speed

In the first set of experiments, we evaluated the rate of convergence success and speed. Each dataset was trained during 500 epochs and the value of the objective function retained. The following subsections refer to each dataset. Here, the figures show the entropy lines for each trial and the corresponding mean curves (the latter in the bottom right figure). In the table we count the percentage of convergence success for several values of entropy and the corresponding mean number of iterations (only for the successful curves).

Iris

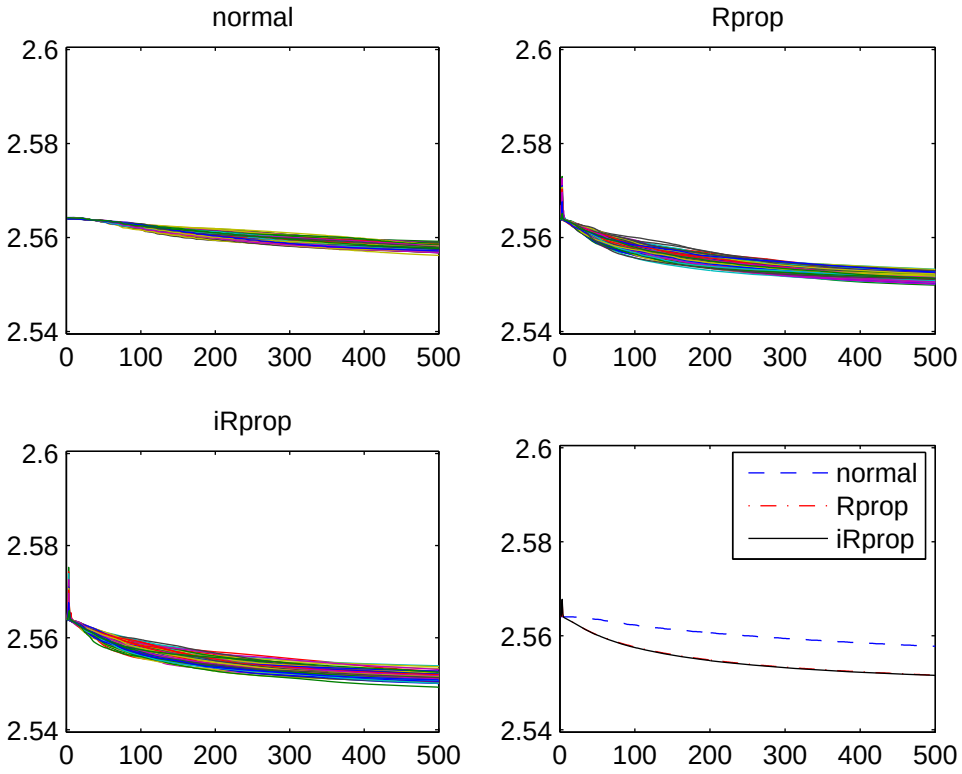


Iris	7.688	7.685	7.682	7.680
Normal	100% - 42	100% - 102	100% - 214	66% - 257
Rprop	100% - 8	100% - 26	99% - 75	99% - 164
iRprop	100% - 9	100% - 28	100% - 83	99% - 172

The mean curves for Rprop and iRprop are significantly better than the one from Normal, which means that a faster trainig is performed by the resilient methods. Although not visible, the Wilcoxon test shows that Rprop is better[1] than iRprop during the initial 100 epochs. We also see greater deviations of the 100 trials from the mean curves for iRprop/Rprop than for Normal. From the table, we also see that a less number of iterations of Rprop/iRprop methods are needed to achieve the convergence values. Normal looses (dramatically) convergence capacity for the last entropy value.

[1]For now on, whenever one says better it means that it is significant by the Wilcoxon test.

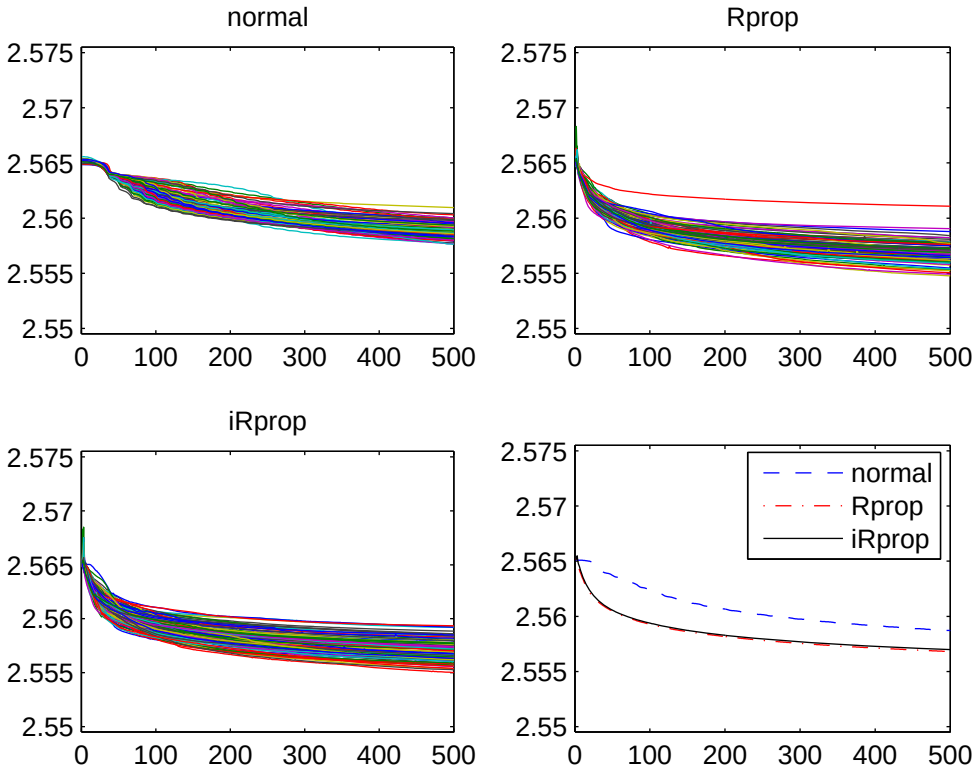
Pima



Pima	2.562	2.560	2.558	2.555
Normal	100% - 117	100% - 257	66% - 280	-
Rprop	100% - 28	100% - 56	100% - 94	100% - 196
iRprop	100% - 28	100% - 55	100% - 92	100% - 191

We found no differences between Rprop and iRprop, while Normal has a significant worst performance when compared to those methods. Note from the table that the resilient methods doesn't loose convergence capacity. On the other hand, Normal already perform badly for 2.558 and in the last value it was not able to converge (while Rprop/iRprop have 100% success). Also, for the first two values of entropy, Normal needs roughly 4 times more epochs than Rprop/iRprop.

Wdbc



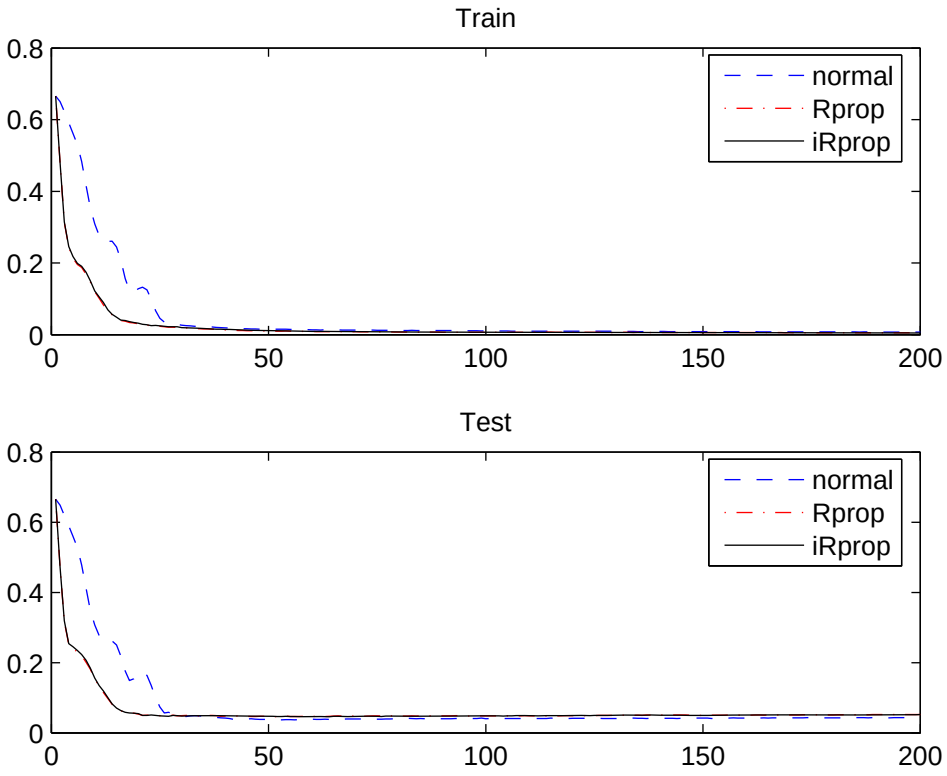
Wdbc	2.565	2.562	2.560	2.558
Normal	100% - 16	100% - 121	95% - 251	7% - 33
Rprop	100% - 3	100% - 24	99% - 70	93% - 209
iRprop	100% - 3	100% - 26	100% - 81	83% - 188

The resilient methods are again better than Normal. The Wilcoxon test also showed that Rprop performs better than iRprop for the first 25 epochs, which is not evident from the figure. From tha table we see the need for a higher number of epochs for Normal ant its degrading behaviour for smaller values of entropy. It is also interesting to see that Rprop is, among the resilient methods, the first to loose convergence capacity (see why in the top right figure), but then iRprop degrades more for the last value of entropy.

1.2.2 Evaluating generalization ability

The second set of experiments was devoted to the evaluation of generalization error of the proposed methods. For each dataset, 70% was used for training and 30% for testing. In each trial, this division was the same for all algorithms. The figures show the mean training and test error curves for three datasets. The tables show the minimumtest error achieved and corresponding epoch. standard deviations are in brackets. The last two columns have the p-values of the Wilcoxon test for comparison of the mean values.

Iris

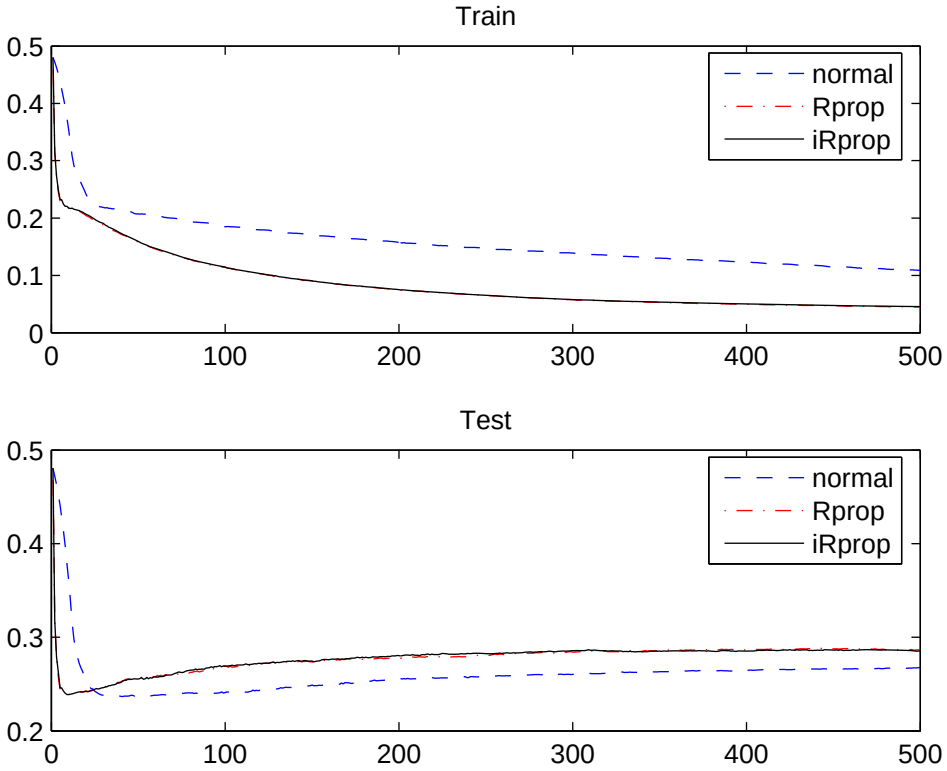


- Train
  Rprop/iRprop are significantly better than Normal mainly in the first 30 epochs, while between them we found no difference.

- Test
  Again iRprop and Rprop perform equally. After the first 30 epochs, Normal generalizes significantly better. Also the minimum value for test error in Normal is different from the other two.

	Test error	epoch	Normal	Rprop
Normal	3.73(2.71)	56		
Rprop	4.58(2.99)	52	0.000	
iRprop	4.58(2.99)	59	0.000	0.6199

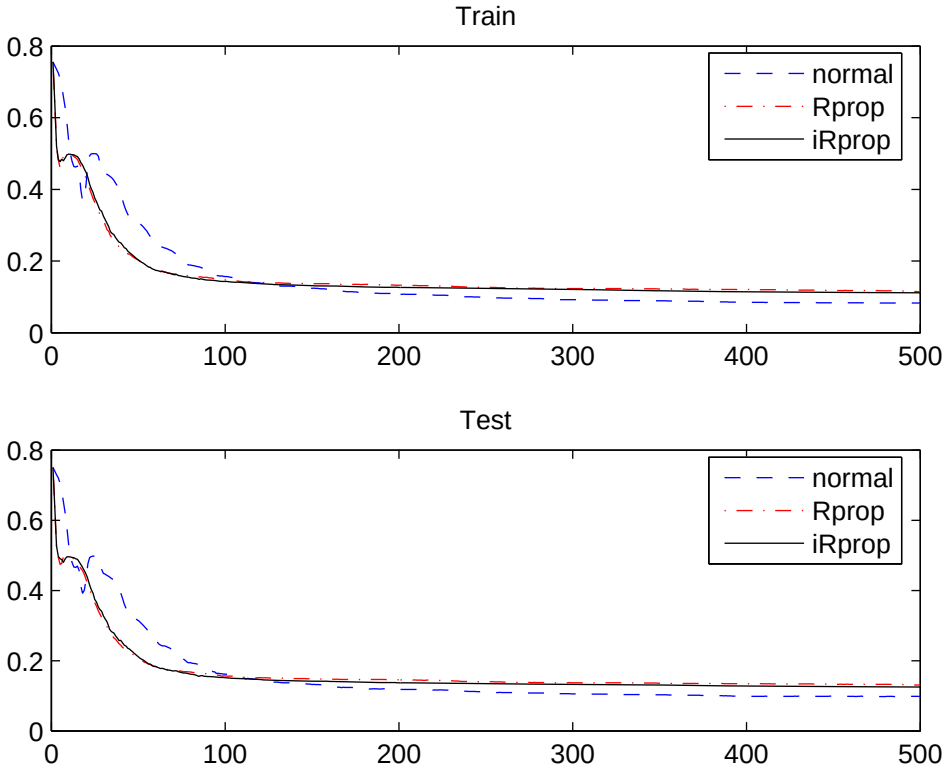
Pima



- Train
  iRprop and Rprop perform equally. Normal is significantly worst than the other two.
- Test
  Rprop and iRprop are very fast to attain the minimum error but overfit very rapidly when compared to Normal. However, when comparing the minimum errors for the three methods we see that they are not significantly different (see the table).

	Test error	epoch	Normal	Rprop
Normal	23.68(2.33)	41		
Rprop	23.87(2.35)	10	0.1686	
iRprop	23.87(2.48)	9	0.1878	0.9154

Pb12



- Train
  Rprop and iRprop perform equally and both are better than Normal. However, this behaviour reverses at the very last epochs.

- Test
  We have a similar behaviour as in training, but here after 250 epochs Normal is significantly better.

	Test error	epoch	Normal	Rprop
Normal	9.79(6.88)	485		
Rprop	13.07(9.70)	500	0.006	
iRprop	12.49(8.03)	497	0.002	0.700

Chapter 2

Resilient methods for Z-EDM

2.1 Resilient methods

In order to speedup the convergence of the proposed algorithm we have selected 3 algorithms from the literature. We present a brief description of each

- Normal
  The first one uses the same strategy as in [1], where a variable learning rate is used to control the descent (in this case the ascent) through the performance surface. In the presence of a decrease of the value of f at the origin from an epoch to another, the network is set to the previous state and the learning rate is decreased, otherwise, η is increased to speedup convergence.

- Rprop
  This method, proposed by Riedmiller and Braun [2, 3], is considered among the best performing first-order learning methods for neural networks. It controls every single weight connection by using individual stepsizes. These stepsizes are updated by only inspecting successive steps of the derivative's signal. Independence from the magnitude of the derivative and equal learning on the entire network are among the benefits of the method cite.

- iRprop
  iRprop stands for Improved Rprop and was proposed by Igel and Husken [4, 5]. In fact, iRprop$^+$ is the improved version used here (see cite for more approaches). The difference to the original Rprop algorithm relies on the combination of individual information about the error surface (sign of the derivative) with more global information (the value of the error itself) to provide a weight-backtracking scheme.

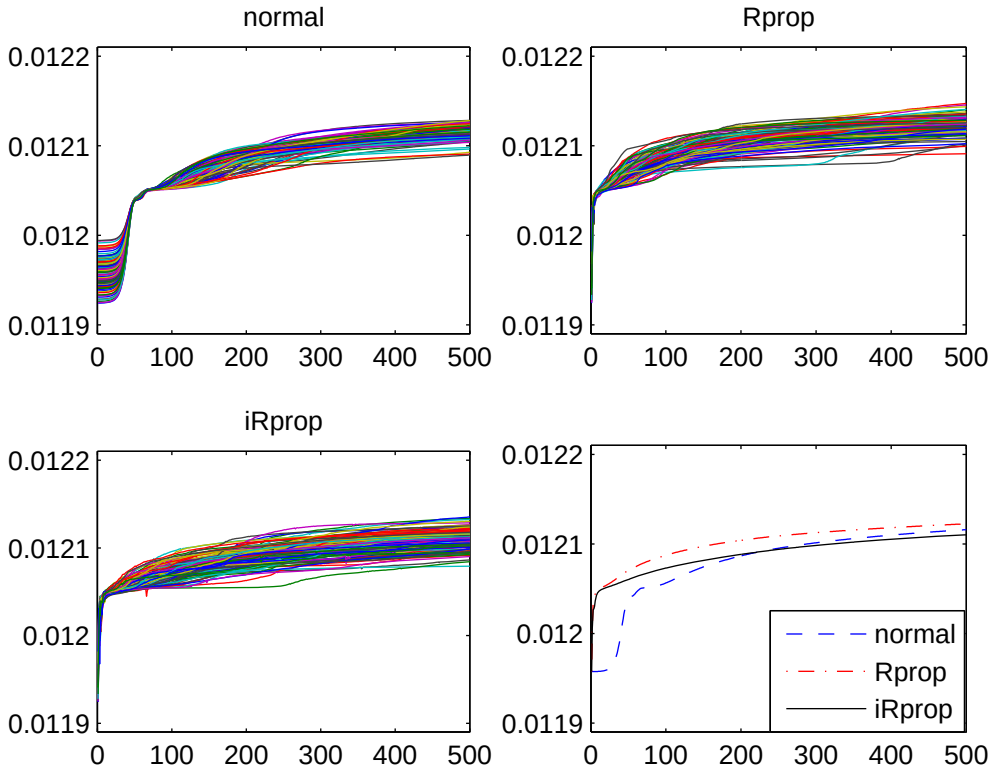
2.2 Experiments and Results

We performed two types of experiments and used 4 well known datasets: Pb12 from [6] and Iris, Wdbc and Pima from the UCI repository [7]. To have a fair comparison, a total of 100 trials of each experiment was made and the initial weights were always the same (for each trial) for each algorithm.

2.2.1 Evaluating convergence speed

In the first set of experiments, we evaluated the rate of convergence success and speed. Each dataset was trained during 500 epochs and the value of the objective function retained. The following subsections refer to each dataset. Here, the figures show the entropy lines for each trial and the corresponding mean curves (the latter in the bottom right figure). In the table we count the percentage of convergence success for several values of entropy and the corresponding mean number of iterations (only for the successful curves).

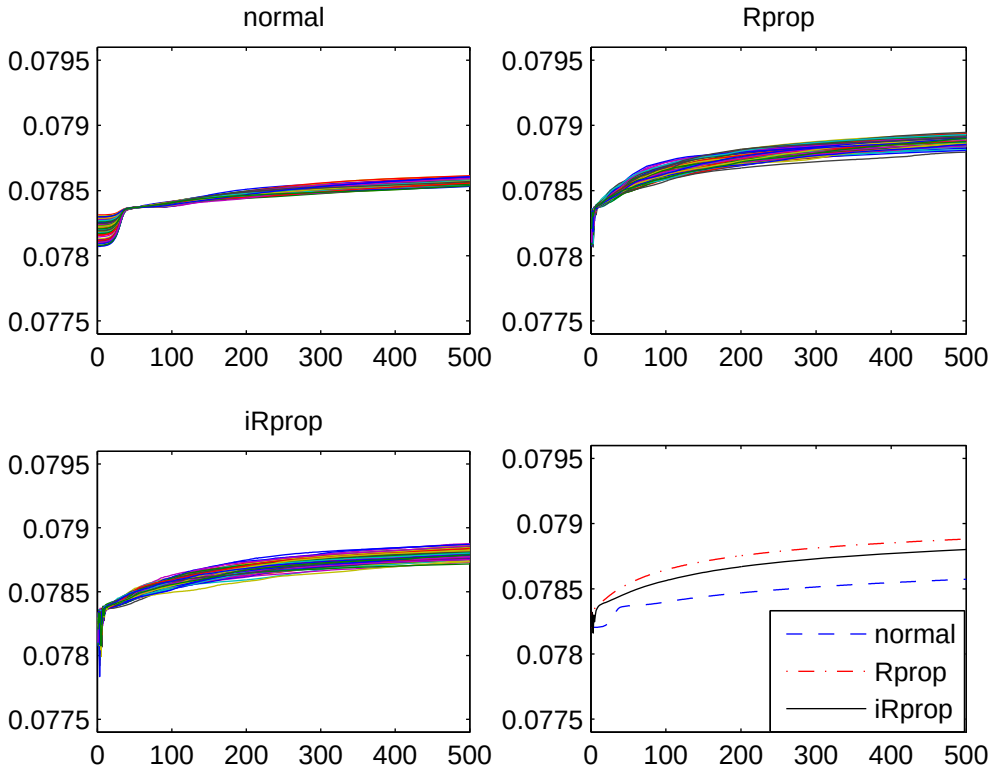
Iris



	0.01209	0.0121	0.01211	0.01212
Normal	99% - 218	95% - 273	82% - 309	38% - 171
Rprop	100% - 128	98% - 181	89% - 228	60% - 195
iRprop	97% - 222	80% - 235	50% - 172	22% - 92

Rprop performs significantly better than the other two methods. While iRprop is better than Normal during the initial 200 epochs, this behaviour reverses in the last epochs where the latter is significantly better. Analyzing the table, one can see the fast degrade of iRprop, while, of course, Rprop is always the bets approach.

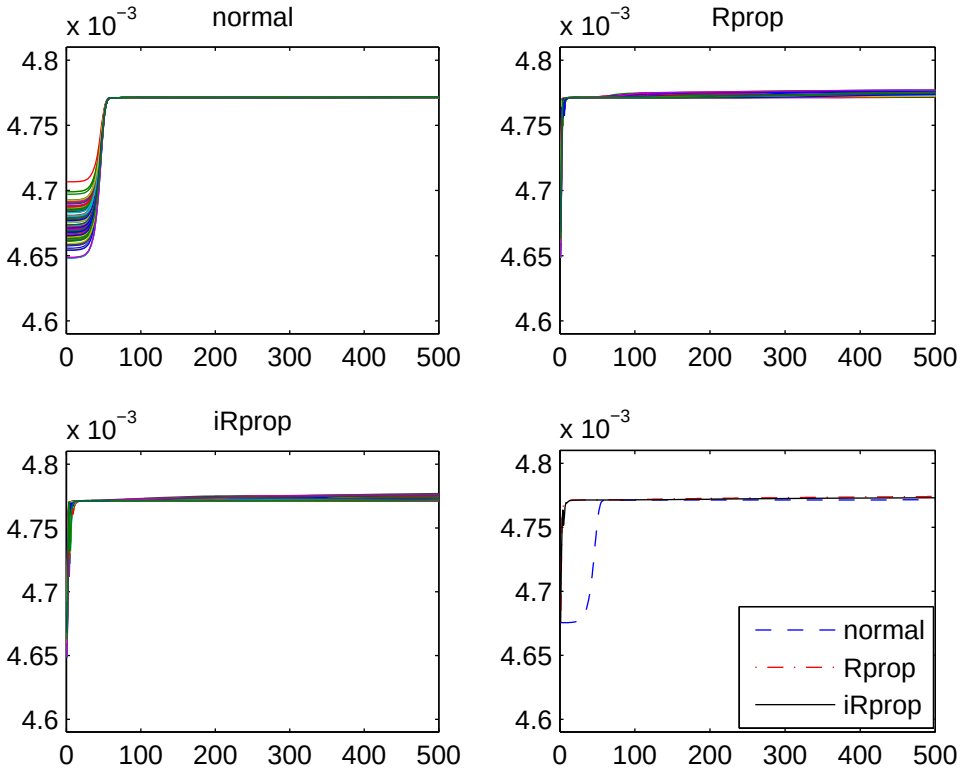
Pima



	0.0785	0.0786	0.0788	0.0789
Normal	100% - 272	5% - 24	-	-
Rprop	100% - 39	100% - 77	99% - 275	31% - 139
iRprop	100% - 65	100% - 131	54% - 230	-

Again, Rprop is always significantly better than the other two methods. However, with this dataset, iRprop is also better than Normal in all the training phase. From the table we also see that Normal looses convergence capacity very soon. In the first value of the objective function (the only one comparable), Rprop needs only 39 epochs (in mean) while Normal needs 272.

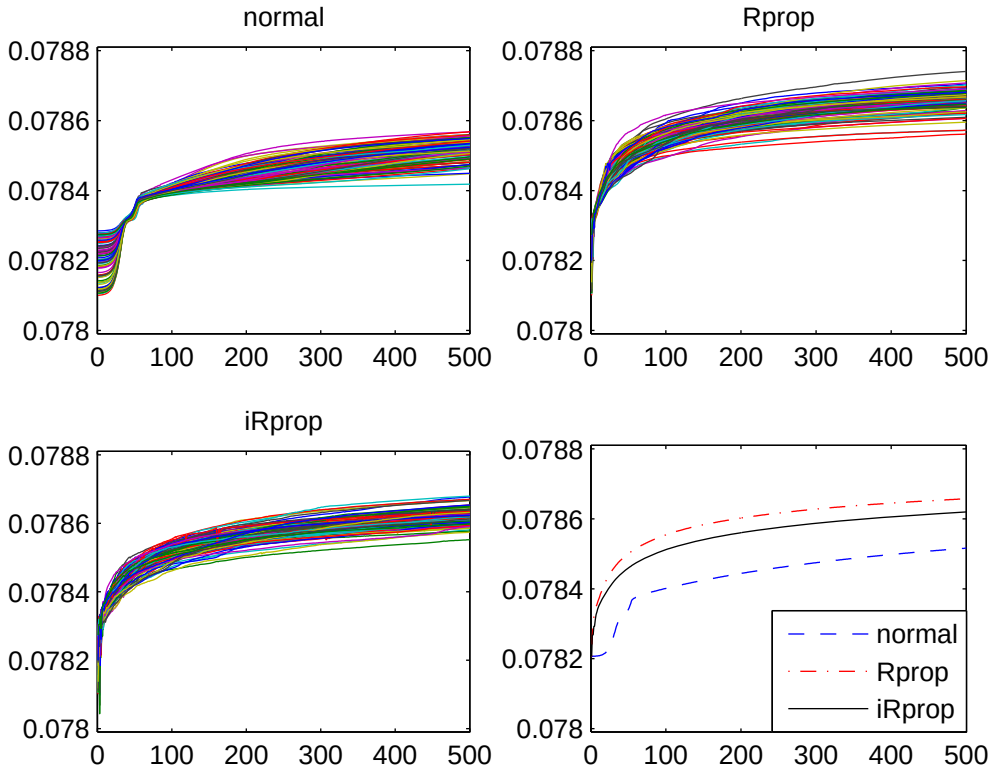
Pb12



	0.004770	0.004772	0.004774	0.004776
Normal	100% - 56	-	-	-
Rprop	100% - 6	97% - 185	53% - 157	10% - 39
iRprop	100% - 9	80% - 205	28% - 87	2% - 9

iRprop and Rprop perform always better than Normal. Although visually the differences after the first 50 epochs are quite small, they are always significant by the Wilcoxon test. The same happens between Rprop and iRprop with advantage for Rprop. Note also from the figures the similar behaviour of all 100 trials in each method

Wdbc



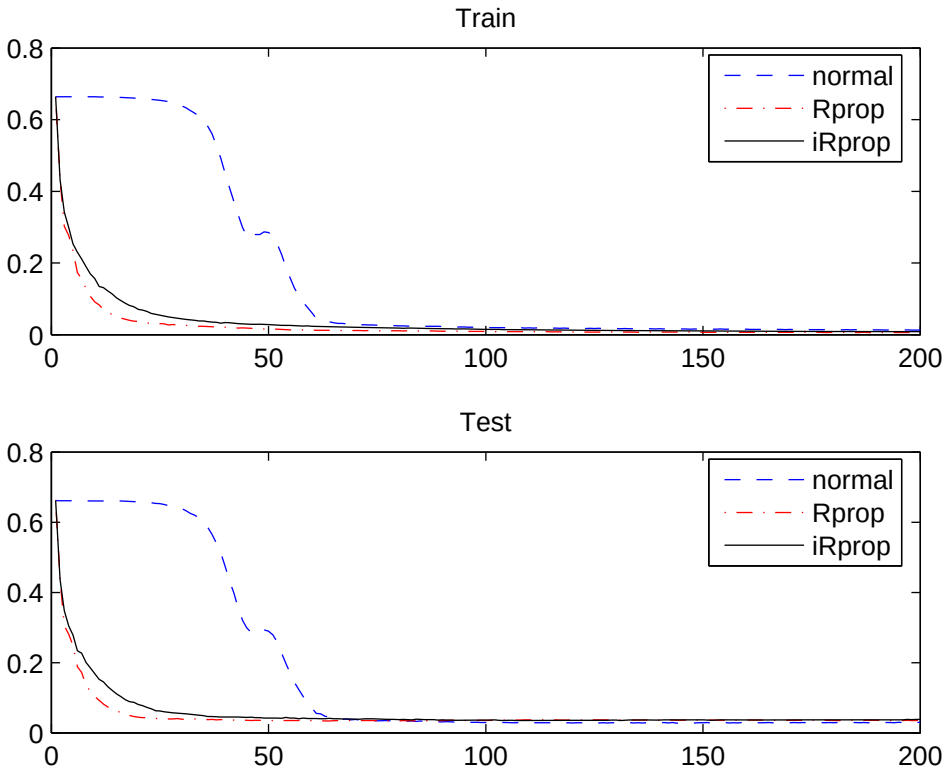
	0.0784	0.0785	0.07855	0.0786
Normal	100% - 104	72% - 254	15% - 70	-
Rprop	100% - 17	100% - 53	100% - 104	96% - 192
iRprop	100% - 25	100% - 90	100% - 183	79% - 269

We found all the differences encountered in the bottom right figure significant by the Wilcoxon test. These are the orders: 1st - Rprop; 2nd - iRprop; 3th - Normal. By the table we also see tha fast degrade of Normal and the difference in terms of mean number of iterations needed: Rprop is approximately 5 times faster than Normal.

2.2.2 Evaluating generalization ability

The second set of experiments was devoted to the evaluation of generalization error of the proposed methods. For each dataset, 70% was used for training and 30% for testing. In each trial, this division was the same for all algorithms. The figures show the mean training and test error curves for three datasets. The tables show the minimumtest error achieved and corresponding epoch. standard deviations are in brackets. The last two columns have the p-values of the Wilcoxon test for comparison of the mean values.

Iris

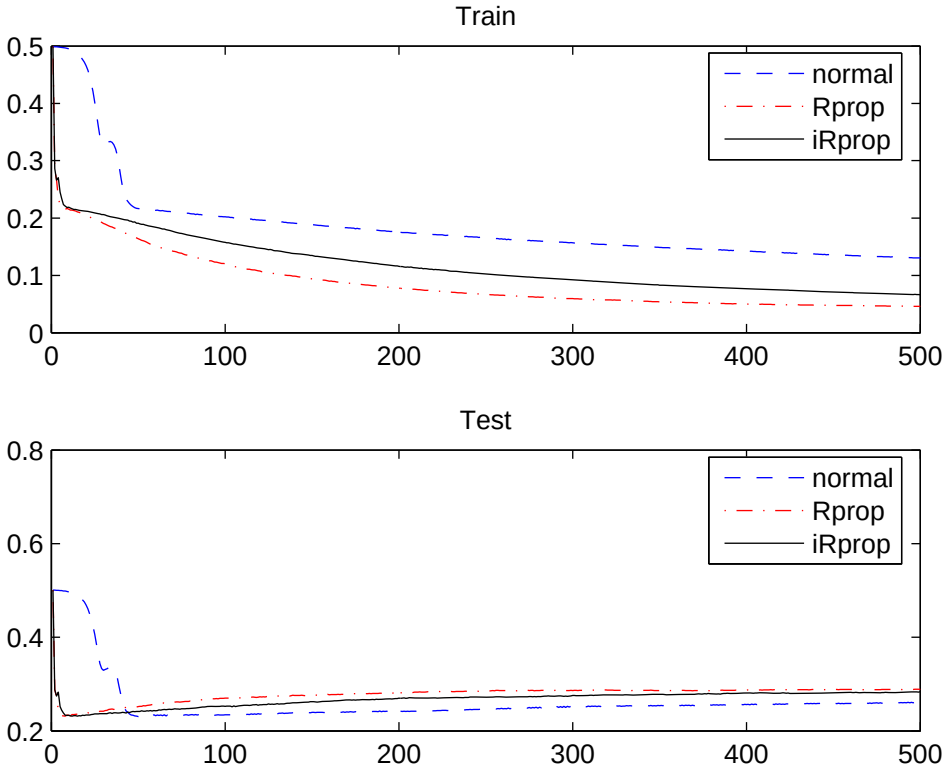


- Train
  Rprop is significantly better than the other two methods in all 200 epochs. It is also very fast at the first epochs.
- Test
  A similar behaviour is encountered here (see the figures) except that after 80 epochs Normal has a significant better generalization performance.

Rprop performs better than iRprop.

	Test error	epoch	Normal	Rprop
Normal	2.78(2.33)	144		
Rprop	3.40(2.57)	80	0.000	
iRprop	3.51(2.65)	1.07	0.000	0.373

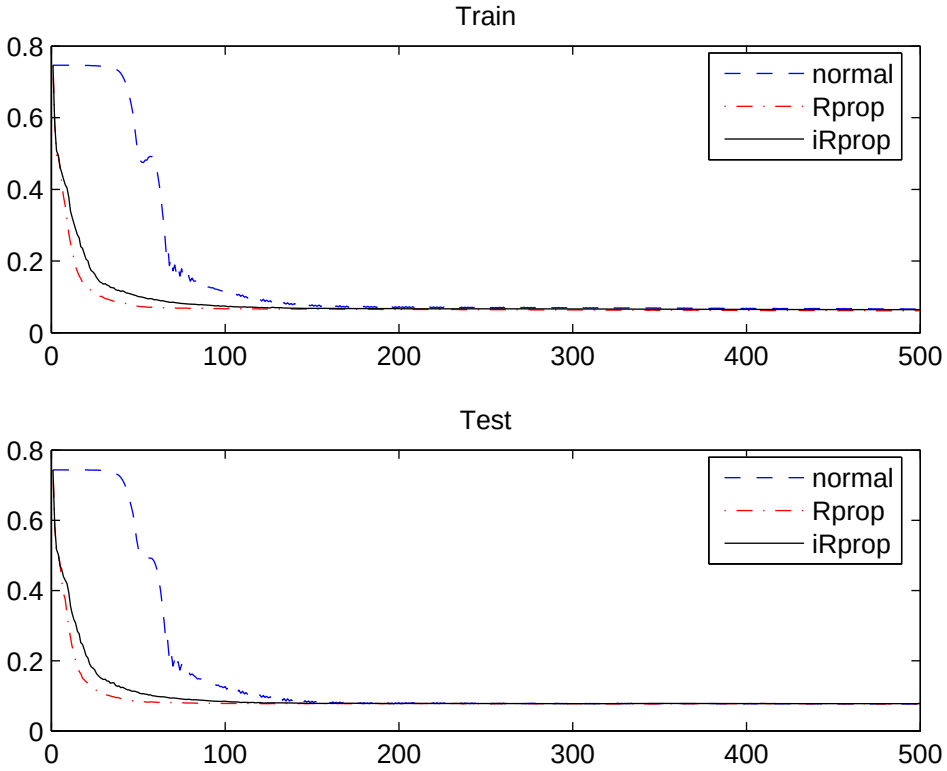
Pima



- Train
  Rprop is significantly better than the other two methods
- Test
  Rprop and iRprop are very fast to attain the minimum error but overfit very rapidly when compared to Normal (iRprop doesn't overfit as rapidly as Rprop). However, when comparing the minimum errors for the three methods we see that they are not significantly different (see the table).

	Test error	epoch	Normal	Rprop
Normal	23.08(2.38)	50		
Rprop	23.15(2.30)	8	0.396	
iRprop	23.14(2.37)	15	0.453	0.998

Pb12

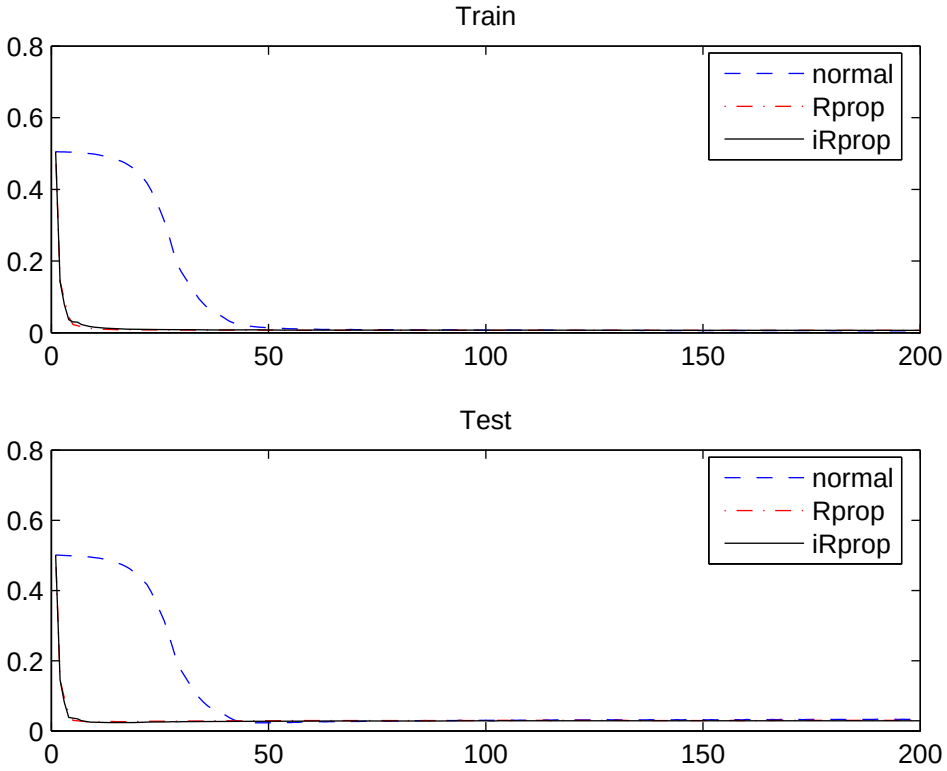


- Train
  Rprop performs significantly better than iRprop and Normal by the Wilcoxon test, although visually the differences are not detected after 150/200 epochs.

- Test
  We have a similar behaviour as before but after (around) 150 epochs, all the methods perform equally by the Wilcoxon test. This test also shows that the minimum test errors achieved are not different.

	Test error	epoch	Normal	Rprop
Normal	7.66(1.62)	497		
Rprop	7.67(1.58)	319	0.599	
iRprop	7.72(1.58)	464	0.275	0.489

Wdbc



- Train
  Rprop and iRprop are faster at the beginning with advantage for Rprop for the first 50 epochs. Only for the last 30 epochs, Normal is better by the Wilcoxon test (which is not visible!)

- Test
  Rprop and iRprop are better than Normal for the first 40 epochs, then Normal is better between 45 to 75 epochs and Rprop and iRprop are again better after around 120 epochs. iRprop is better than Rprop between 15 and 30 epochs. From the table one can see that the minimum of Normal and Rprop are different, while Normal and iRprop are equal.

	Test error	epoch	Normal	Rprop
Normal	2.30(1.01)	49		
Rprop	2.54(1.08)	11	0.008	
iRprop	2.35(1.05)	17	0.557	0.013

Bibliography

[1] L.M. Silva, J. Marques de Sá, and L.A. Alexandre. Neural Network Classification using Shannon's Entropy. In *European Symposium on Artificial Neural Networks*, 2005.

[2] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Proc. of the IEEE Intl. Conf. on Neural Networks*, pages 586–591, San Francisco, CA, 1993.

[3] M. Riedmiller and H. Braun. Rprop – description and implementation details, 1994.

[4] Christian Igel and Michael Hüsken. Improving the Rprop learning algorithm, 2000.

[5] C. Igel and M. Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

[6] R. Jacobs, M. Jordan, S. Nowlan, and G. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[7] C.L. Blake and C.J. Merz. UCI Repository of machine learning databases. University of California, Irvine, Dept. of Information and Computer Sciences, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.