Study of Instance Selection Methods

Seminar at Aston University

Álvar Arnaiz González

May 21, 2018

Universidad de Burgos

Table of contents

1. First of all

Who are we?

Researchers in Universidad de Burgos

Research group: 'ADMIRABLE' (Advanced Data MIning Research And (Business intelligence | Bioinformatics | Big Data) LEarning – http://admirable-ubu.es/)

- Álvar Arnaiz González
  - PhD on Computer Science (Universidad de Burgos)
- César García Osorio
  - PhD on Computer Science (University of The West of Scotland)
- Carlos López Nozal
  - PhD on Computer Science (Universidad de Valladolid)
- Mario Juez Gil
  - PhD candidate on Computer Science (Universidad de Burgos)

Where is Burgos?[1]

[1]Copyright by NordNordWest - own work, using United States National Imagery and Mapping Agency data, CC BY-SA 3.0

Why is Burgos famous for? (i)[2]

[2]Copyright by Camino del Cid - Own work, CC BY-SA 3.0

Why is Burgos famous for? (ii)[3]

[3] Copyright by Sueños de aire azul - Own work

Why is Burgos famous for? (iii)[4]

But it also has got this!

[4] Copyright by Universidad de Burgos - Own work

2. Introduction

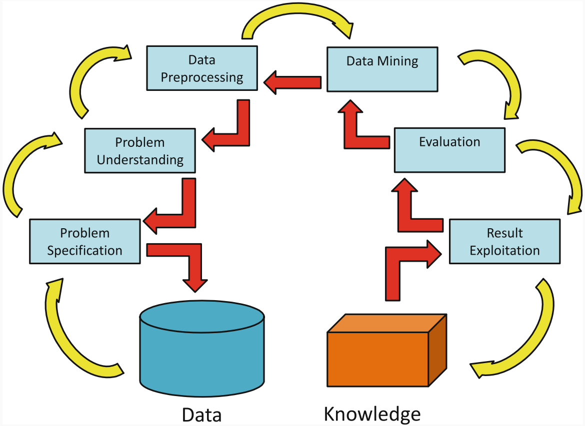


Figure 1: KDD process. Picture reproduced from[5].

[5] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

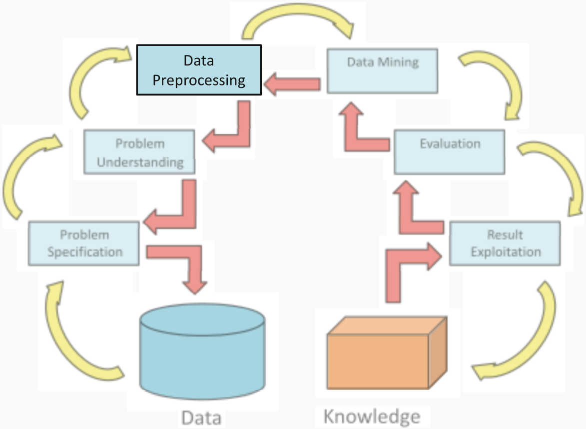


Figure 1: KDD process. Picture reproduced from[5].

[5] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

Data Preprocessing

Data preparation: set of techniques that initialize the data properly to serve as an input for a certain DM algorithm.

- Data cleaning.
- Data transformation.
- Data integration.
- Data normalization.
- Missing data imputation.
- Noise identification.

Data reduction: set of techniques that obtain a reduced representation of the original data.

- Discretization.
- Feature selection.
- Instance selection.

Data preparation: set of techniques that initialize the data properly to serve as an input for a certain DM algorithm.

- Data cleaning.
- Data transformation.
- Data integration.
- Data normalization.
- Missing data imputation.
- Noise identification.

Data reduction: set of techniques that obtain a reduced representation of the original data.

- Discretization.
- Feature selection.
- **Instance selection**.

What instance selection is?

- Its aim: the selection of a subset of instances.
- One restriction: keeping, or even improving, the prediction capabilities of the whole data set.

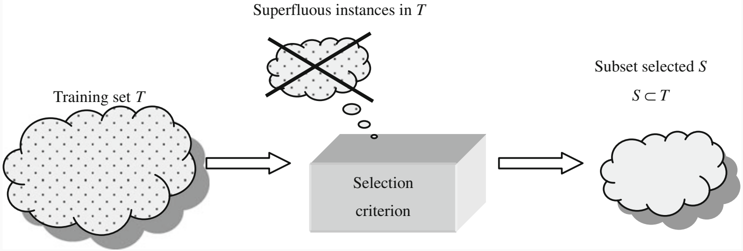
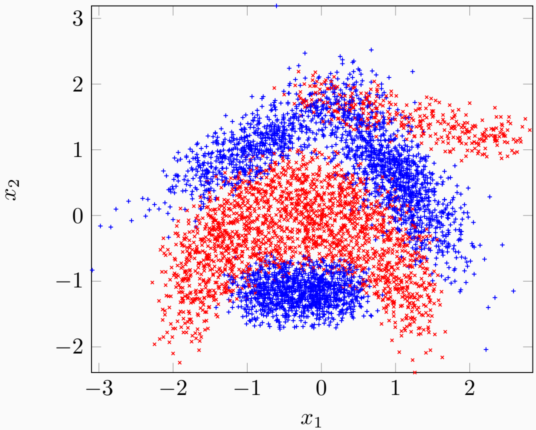


Figure 2: Instance selection process. Picture reproduced from[6].

[6] J. Arturo Olvera-López et al. “A review of instance selection methods”. In: *Artificial Intelligence Review* 34.2 (2010), pp. 133–143. ISSN: 1573-7462.

An example (i)

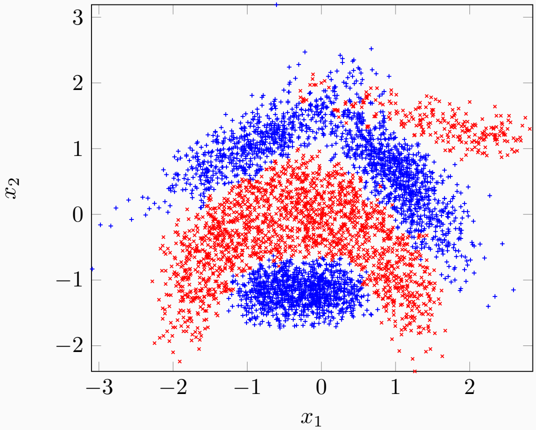
Banana[7] data set (original): 5 300 instances.



[7] Fraunhofer Institute for Intelligent Analysis and Information Systems. *Benchmark Repository*. URL: `http://www.iais.fraunhofer.de/`.

An example (ii)

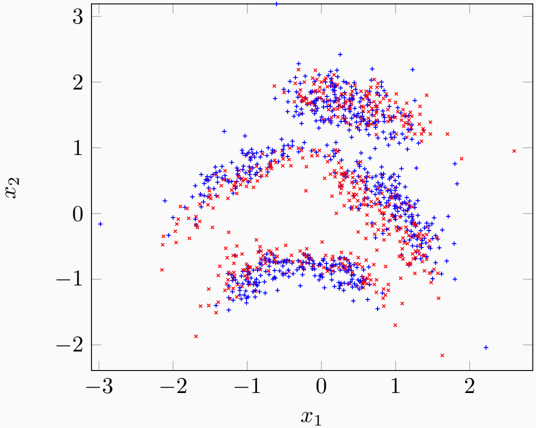
After ENN or *Wilson Editing*[8]: 4 696 inst.



[8] Dennis L. Wilson. "Asymptotic Properties of Nearest Neighbor Rules Using Edited Data". In: *Systems, Man and Cybernetics, IEEE Transactions on* SMC-2.3 (1972), pp. 408–421. ISSN: 0018-9472.

An example (iii)

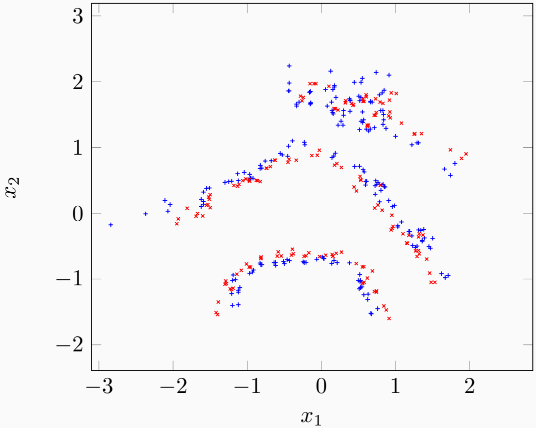
After *Condensed Nearest Neighbour*[9] (CNN): 1 199 inst.



[9]P. Hart. "The condensed nearest neighbor rule (Corresp.)" In: *Information Theory, IEEE Transactions on* 14.3 (1968), pp. 515 –516. ISSN: 0018-9448.

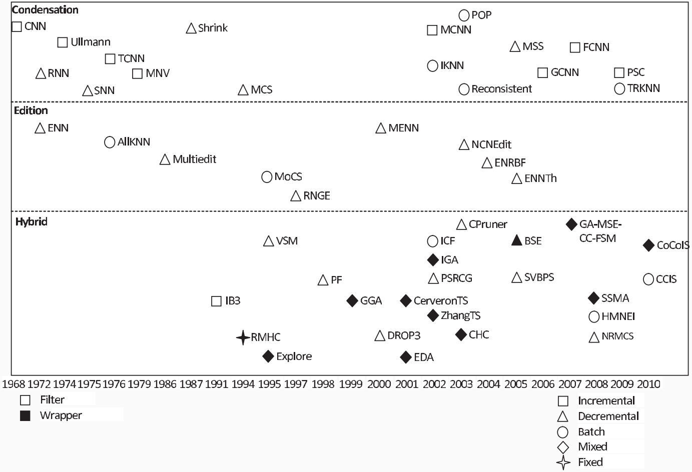
An example (iv)

After *Decr. Red. Optimization Procedure* 3 (DROP3)[10]: 350 inst.



[10] D.Randall Wilson and Tony R. Martinez. "Reduction Techniques for Instance-Based Learning Algorithms". English. In: *Machine Learning* 38.3 (2000), pp. 257–286. ISSN: 0885-6125.

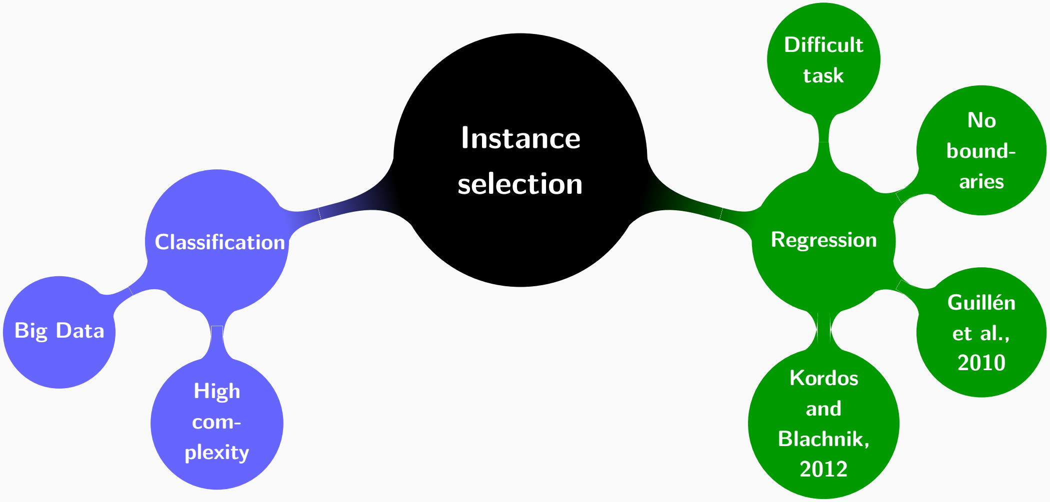
The roadmap of instance selection[11]



[11] S. Garcia et al. “Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.3 (2012), pp. 417–435. ISSN: 0162-8828.

3. Motivation and goals

Instance selection: challenges



Instance selection for regression

- Instance selection for classification has been broadly researched.

Instance selection for regression

- Instance selection for classification has been broadly researched.
- However, instance selection for regression has not, due to its difficulties.

Instance selection for regression

- Instance selection for classification has been broadly researched.
- However, instance selection for regression has not, due to its difficulties.
- There are not well-defined boundaries between classes.

Instance selection for regression

- Instance selection for classification has been broadly researched.
- However, instance selection for regression has not, due to its difficulties.
- There are not well-defined boundaries between classes.
- Two journal papers faced this issue:[12],[13].

[12] Álvar Arnaiz-González et al. "Instance selection for regression by discretization". In: *Expert Systems with Applications* 54 (2016), pp. 340 –350. ISSN: 0957-4174.

[13] Álvar Arnaiz-González et al. "Instance selection for regression: Adapting DROP". . In: *Neurocomputing* 201 (2016), pp. 66 –81. ISSN: 0925-2312.

Combination of instance selection methods

- The benefits of *ensembles* of classifiers/regressors are well-known.

Combination of instance selection methods

- The benefits of *ensembles* of classifiers/regressors are well-known.
- Even *ensembles* of instance selection for classification[14].

[14] Nicolás García-Pedrajas and Aida de Haro-García. “Boosting instance selection algorithms”. In: *Knowledge-Based Systems* 67 (2014), pp. 342 –360. ISSN: 0950-7051.

Combination of instance selection methods

- The benefits of *ensembles* of classifiers/regressors are well-known.
- Even *ensembles* of instance selection for classification[14].
- However, *ensembles* of instance selection for regression had not been tested before.

[14] Nicolás García-Pedrajas and Aida de Haro-García. "Boosting instance selection algorithms". In: *Knowledge-Based Systems* 67 (2014), pp. 342 –360. ISSN: 0950-7051.

Combination of instance selection methods

- The benefits of *ensembles* of classifiers/regressors are well-known.
- Even *ensembles* of instance selection for classification[14].
- However, *ensembles* of instance selection for regression had not been tested before.
- A journal paper faced this issue:[15].

[14] Nicolás García-Pedrajas and Aida de Haro-García. "Boosting instance selection algorithms". In: *Knowledge-Based Systems* 67 (2014), pp. 342 –360. ISSN: 0950-7051.
[15] Álvar Arnaiz-González et al. "Fusion of instance selection methods in regression tasks". In: *Information Fusion* 30 (2016), pp. 69 –79. ISSN: 1566-2535.

- The computational complexity of instance selection methods is commonly very high.

Instance selection for big data

- The computational complexity of instance selection methods is commonly very high.
- This makes its use almost impossible for huge data sets

Instance selection for big data

- The computational complexity of instance selection methods is commonly very high.
- This makes its use almost impossible for huge data sets
- Some scale up approaches based on divide-and-conquer have emerged.

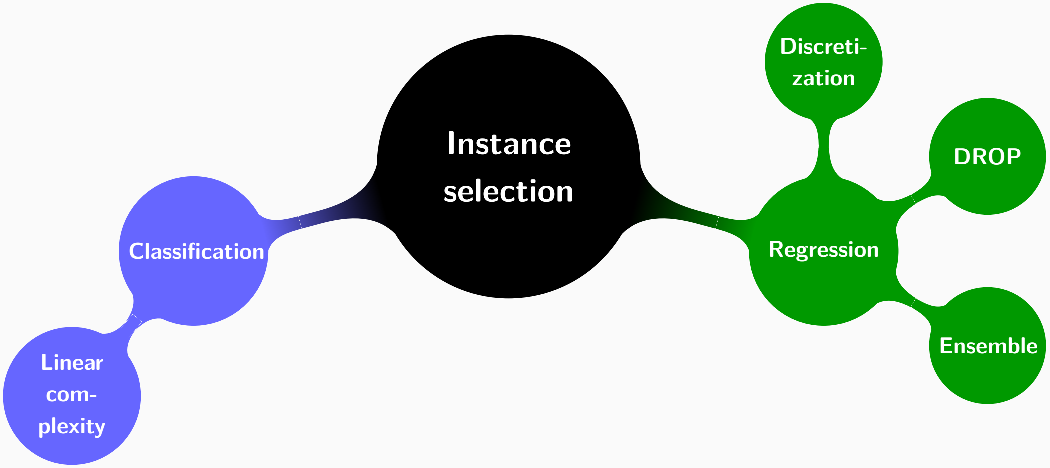
Instance selection for big data

- The computational complexity of instance selection methods is commonly very high.
- This makes its use almost impossible for huge data sets
- Some scale up approaches based on divide-and-conquer have emerged.
- We propose a method that, without being based on divide and conquer, is capable of achieving a linear complexity.

Instance selection for big data

- The computational complexity of instance selection methods is commonly very high.
- This makes its use almost impossible for huge data sets
- Some scale up approaches based on divide-and-conquer have emerged.
- We propose a method that, without being based on divide and conquer, is capable of achieving a linear complexity.
- The key to the method is in the use it makes of *locality sensitive hashing* (LSH)[16].

[16] Álvar Arnaiz-González et al. "Instance selection of linear complexity for big data". In: *Knowledge-Based Systems* 107 (2016), pp. 83 –95. ISSN: 0950-7051.


Instance selection
Classification
Linear com-plexity
Regression
Discreti-zation
DROP
Ensemble

4. Instance selection for regression by discretization

Discretization

The idea

Discretize the numeric class and apply a well-known IS method for classification

Meta-model proposed

Algorithm 1: Proposed meta-model based on discretization of the output variable

Input: Training set $\{\mathbf{X}, Y\} = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$, Discretization algorithm and all the parameters that it needs

Output: Instance set $S \subseteq \{\mathbf{X}, Y\}$

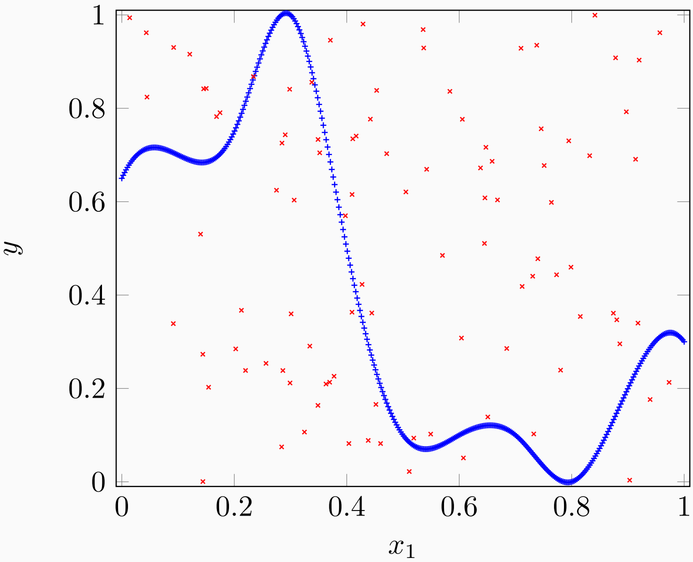
$Y_D =$ Discretization of the numerical target Y

Apply classification-based instance selection algorithm over $\{\mathbf{X}, Y_D\}$ to obtain subset S

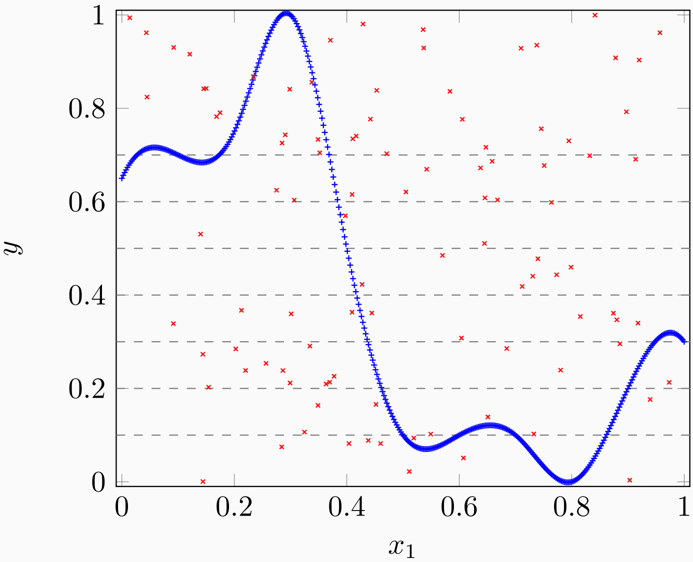
Restore the numerical value of the output variable in S

return S

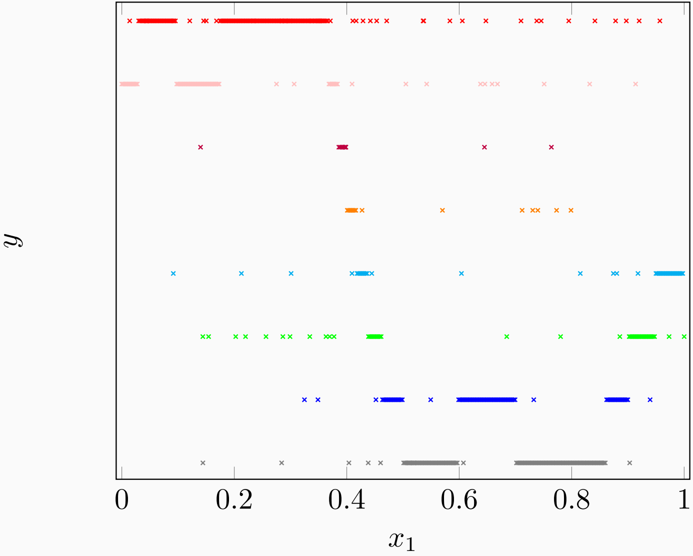
How does it work? An example (i)



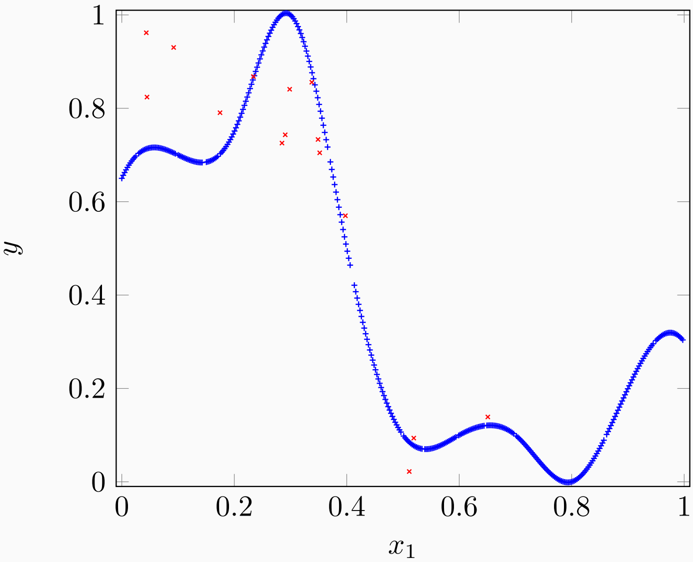
How does it work? An example (ii)



How does it work? An example (iii)



How does it work? An example (iv)



Experimental setup: schematic view

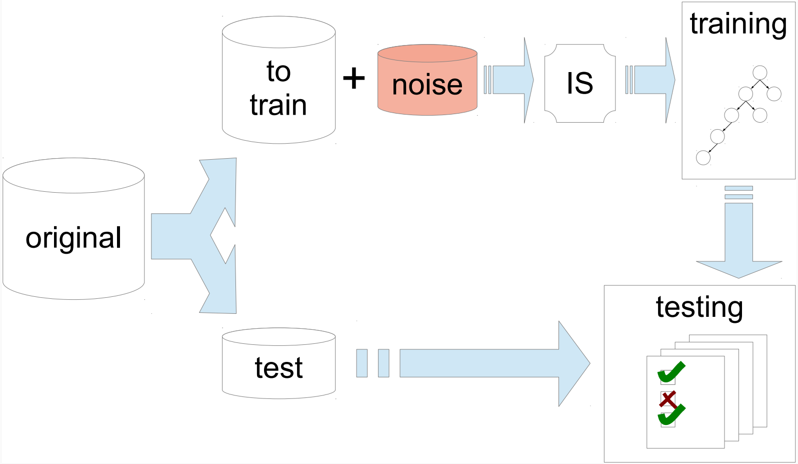


Figure 3: Configuration of the experiments.

Data sets

Data set	# Instances	# Attributes	RMSE		
			*k*NN	RBF	REPTree
machineCPU	209	6	73.3861	54.9182	74.1478
baseball	337	16	681.9675	694.5271	784.2473
dee	365	6	0.4136	0.4024	0.4886
autoMPG8	392	7	2.9245	2.6252	3.2893
autoMPG6	392	5	2.7766	2.9610	3.2841
ele-1	495	2	647.7872	637.9696	709.1737
stock	950	9	0.7816	1.0196	1.1843
laser	993	4	10.2126	7.4007	14.0605
concrete	1 030	8	9.3890	7.2785	7.4055
treasury	1 049	15	0.2423	0.2265	0.3214
mortgage	1 049	15	0.1917	0.1063	0.2562
ele-2	1 056	4	271.1403	123.7133	185.0631
friedman	1 200	5	1.7855	1.5425	2.7496
wizmir	1 461	9	1.7195	1.1542	1.7374
wankara	1 609	9	1.9401	1.2973	2.0441
plastic	1 650	2	1.6412	1.5113	1.7518
quake	2 178	3	0.1954	0.1887	0.1887
ANACALT	4 052	7	0.1188	0.1889	0.0709
abalone	4 177	8	2.2223	2.0983	2.3359
delta-ail	7 129	5	0.0002	0.0002	0.0002
compactiv	8 192	21	3.0811	3.5825	3.2458
puma32h	8 192	32	0.0273	0.0232	0.0089
delta-elv	9 517	6	0.0015	0.0014	0.0015
ailerons	13 750	40	0.0002	0.0002	0.0002
pole	14 998	26	8.2376	16.7730	7.1492
elevators	16 599	18	0.0036	0.0022	0.0036
california	20 640	8	61915.5979	62456.4909	58826.3442
house	22 784	16	38444.1767	38512.3930	38854.6220
mv	40 768	10	1.8591	0.6156	0.3047

Experimental setup: regressors and IS

Three regressors were used: kNN, REPTree and RBF.

The proposed meta-model was compared against:

- Mutual information (MI)[17]: $k = 6$ and $\alpha = 0.05$.
- ENN based on threshold (RegENN)[18]: $k = 9$ and $\alpha = 5$.

Noise levels: 10%, 20%, 30%, and 40%, adding or subtracting a random value to the target attribute.

[17] A. Guillen et al. "New method for instance or prototype selection using mutual information in time series prediction". In: *Neurocomputing* 73.10-12 (2010). Subspace Learning / Selected papers from the European Symposium on Time Series Prediction, pp. 2030 –2038. ISSN: 0925-2312.

[18] Mirosław Kordos and Marcin Blachnik. "Instance selection with neural networks for regression problems". In: *Proceedings of the 22nd international conference on Artificial Neural Networks and Machine Learning - Volume Part II*. ICANN'12. Lausanne, Switzerland: Springer-Verlag, 2012, pp. 263–270. ISBN: 978-3-642-33265-4.

Experimental setup: measures

Confusion matrix:

Predicted	**Actual**	
	Noise	No noise
Noise	TP	FP
No noise	FN	TN

Using it we calculated F_1 *score*:

$$F_1 = 2 \cdot \frac{\texttt{precision} \cdot \texttt{recall}}{\texttt{precision} + \texttt{recall}} \tag{1}$$

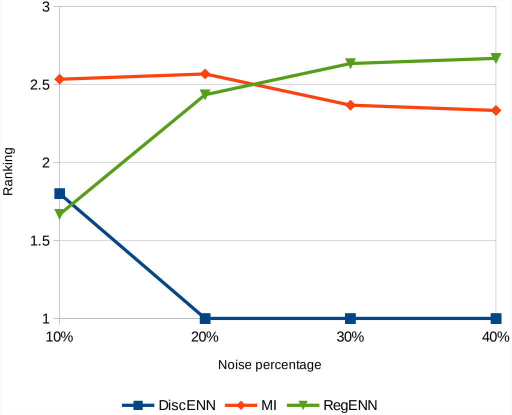
And *G mean*:

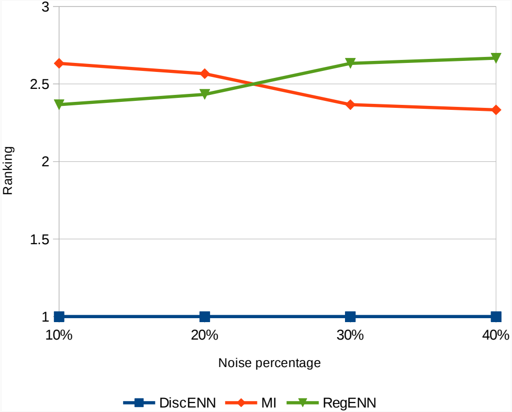
$$G\ mean = \sqrt{\texttt{specifity} \cdot \texttt{recall}} \tag{2}$$

where $\texttt{specifity} = TN/(TN + FP)$, $\texttt{precision} = TP/(TP + FP)$ and $\texttt{recall} = TP/(TP + FN)$.

Results: F_1 and *G mean*

Average ranks over F_1 score (left) and *G mean* (right).





Results: average ranks over RMSE

*k*NN

IS Algorithm	% noise			
	10	20	30	40
DiscENN	2.172	**1.465**	**1.172**	**1.207**
MI	2.534	2.638 ✖	2.207 ✖	2.086 ✖
RegENN	**1.776**	2.224 ✖	3.965 ✖	3.965 ✖
NoFilter	3.517 ✖	3.672 ✖	2.655 ✖	2.741 ✖

RBF

IS Algorithm	% noise			
	10	20	30	40
DiscENN	2.327	**1.672**	**1.431**	**1.465**
MI	2.483	2.603 ✖	2.707 ✖	2.758 ✖
RegENN	**1.983**	2.121	2.327 ✖	2.379 ✖
NoFilter	3.207 ✖	3.603 ✖	3.534 ✖	3.396 ✖

REPTree

IS Algorithm	% noise			
	10	20	30	40
DiscENN	2.172	**1.638**	**1.431**	**1.396**
MI	2.621	2.862 ✖	2.810 ✖	2.931 ✖
RegENN	**2.000**	2.034	2.172 ✖	2.293 ✖
NoFilter	3.207 ✖	3.465 ✖	3.586 ✖	3.379 ✖

Results: compression

Average ranks and Hochberg procedure over compression.

IS Algorithm	% noise			
	10	**20**	**30**	**40**
DiscENN	**1.000**	**1.000**	**1.000**	**1.000**
MI	2.414 ✖	2.414 ✖	2.345 ✖	2.345 ✖
RegENN	2.586 ✖	2.586 ✖	2.655 ✖	2.655 ✖

Conclusions

- Simple idea: easy to implement.
- Meta-model: it can be used with any IS algorithm for classification.
- Competitive results as noise filter.

5. Fusion of instance selection methods in regression tasks

Ensembles

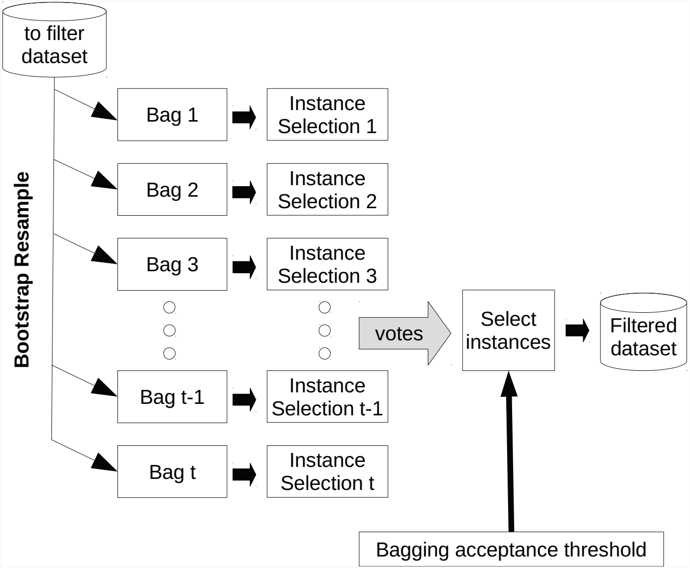
Ensembles have been successfully applied to several problems.

The ensembles' hypothesis claims that the combination of classifiers or regressors performs better than the base methods alone.

The idea

Create ensembles of instance selection methods and test them against the instance selection methods alone.

Schematic view of the IS bagging process



Algorithm 2: ISBagging - Instance Selection Bagging

Input:

- Training set $T = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$
- Instance selection algorithm `ISAlg`
- Number of bags t
- Percent of instances in the bootstrapped training subsets p
- Threshold z

Output: Instance set $P \subseteq T$

```
for i = 1...t do
    S_t = Bootstrap(T, p)
    P_t = ISAlg(S_t)
    v = CollectVotes(P_t, v)
end
P = SelectInstancesByVotes(T, v, z)
return P
```

Experimental setup

Regressor: kNN.

Instance selection algorithms tested:

- Threshold-based[19] ENN and CNN: T-ENN and T-CNN.
- Discretization-based[20] ENN and CNN: D-ENN and D-CNN.

Each algorithm was tested alone and into ensemble: 8 combinations.

---

[19] Mirosław Kordos and Marcin Blachnik. "Instance selection with neural networks for regression problems". In: *Proceedings of the 22nd international conference on Artificial Neural Networks and Machine Learning - Volume Part II*. ICANN'12. Lausanne, Switzerland: Springer-Verlag, 2012, pp. 263–270. ISBN: 978-3-642-33265-4.

[20] Álvar Arnaiz-González et al. "Instance selection for regression by discretization". In: *Expert Systems with Applications* 54 (2016), pp. 340 –350. ISSN: 0957-4174.

Data sets

Dataset	Instances	Attributes			
		Total	real	integer	nominal
diabetes	43	2	2	0	0
machineCPU	209	6	0	6	0
baseball	337	16	2	14	0
dee	365	6	6	0	0
autoMPG8	392	7	2	5	0
autoMPG6	392	5	2	3	0
ele-1	495	2	1	1	0
forestFires	517	12	7	5	0
stock	950	9	9	0	0
laser	993	4	4	0	0
concrete	1 030	8	7	1	0
treasury	1 049	15	15	0	0
mortgage	1 049	15	15	0	0
ele-2	1 056	4	4	0	0
friedman	1 200	5	5	0	0
wizmir	1 461	9	9	0	0
wankara	1 609	9	9	0	0
plastic	1 650	2	2	0	0
quake	2 178	3	2	1	0
ANACALT	4 052	7	7	0	0
abalone	4 177	8	7	1	0
compactiv	8 192	21	21	0	0
tic	9 822	85	0	85	0
ailerons	13 750	40	36	4	0
pole	14 998	26	26	0	0
elevators	16 599	18	14	4	0
california	20 640	8	3	5	0
house	22 784	16	10	6	0

Results: measures

Instance selection is a multi-objective problem.

For joining into a single measure:

$$BF_\gamma = \gamma \cdot \mathit{Accuracy} + (1-\gamma) \cdot \mathit{Compression} \tag{3}$$

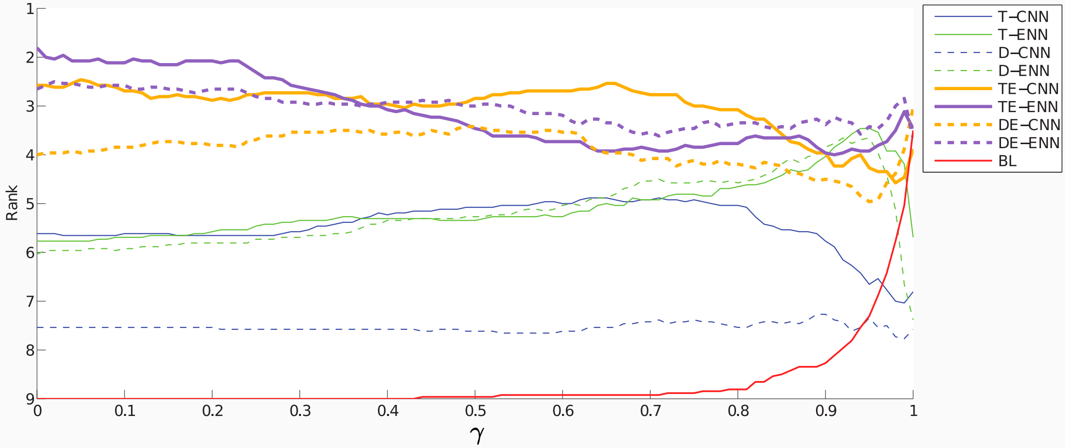
For accuracy: *correlation coefficient* was used.

For compression:

$$C = 1 - \frac{|\texttt{instances after selection}|}{|\texttt{instances before selection}|} \tag{4}$$

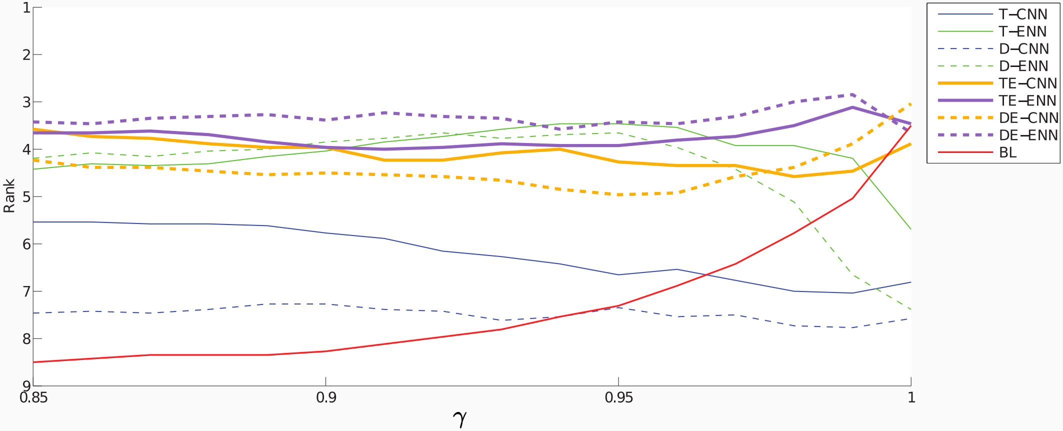
Results: Benefit Function

Average ranks over the benefit function.



Results: Benefit Function (zoom)

Zoom: $\gamma \in [0.85 - 1]$



Results: correlation coefficient

Average ranks and Hochberg procedure over correlation coefficient.

IS algorithm	Ranking	p Hoch.
DE-CNN	3.04	
TE-ENN	3.46	0.577
Baseline	3.50	0.577
DE-ENN	3.65	0.577
TE-CNN	3.88	0.577
T-ENN	5.69	2.38E-3
T-CNN	6.80	4.17E-6
D-ENN	7.38	7.37E-8
D-CNN	7.57	1.84E-8

Results: compression

Average ranks and Hochberg procedure over compression.

IS algorithm	Ranking	p Hoch.
TE-ENN	1.81	
TE-CNN	2.58	0.257
DE-ENN	2.65	0.257
DE-CNN	4.00	3.75E-3
T-CNN	5.62	8.34E-8
T-ENN	5.77	2.75E-8
D-ENN	6.04	2.84E-9
D-CNN	7.54	2.31E-16

Results: ensemble vs. base

The table shows:

- wins/losses (w/l): according to benefit function.
- Wilcoxon test: ✓ indicates that the ensemble method is significantly better than the base method.

Algorithms	γ						
	0.0	**0.25**	**0.50**	**0.70**	**0.80**	**0.90**	**1.00**
TE-ENN vs. T-ENN	26 / 0 ✓	22 / 4 ✓	18 / 8 ✓	17 / 9 ✓	16 / 10 =	13 / 13 =	22 / 4 ✓
TE-CNN vs. T-CNN	25 / 1 ✓	26 / 0 ✓	25 / 1 ✓	25 / 1 ✓	23 / 3 ✓	21 / 5 ✓	25 / 1 ✓
DE-ENN vs. D-ENN	26 / 0 ✓	22 / 4 ✓	22 / 4 ✓	15 / 11 ✓	15 / 11 ✓	16 / 10 =	24 / 2 ✓
DE-CNN vs. D-CNN	26 / 0 ✓	26 / 0 ✓	25 / 1 ✓	25 / 1 ✓	24 / 2 ✓	24 / 2 ✓	25 / 1 ✓

Conclusions

- Ensembles give better results than the base methods alone.
- Versatility: the threshold (z) guides the performance of the IS.
  - More accuracy.
  - More reduction.
- Easy to parallelize: each IS execution is independent to others.

6. Instance selection for regression: adapting DROP

Decremental Reduction Optimization Procedure[21] is a family of IS algorithms.

It consists of 5 algorithms: DROP1, DROP2, DROP3, DROP4, and DROP5.

DROP3 is one of the best IS methods for classification[22].

The idea

Adapt the family of DROP algorithms to regression

---

[21] D.Randall Wilson and Tony R. Martinez. "Reduction Techniques for Instance-Based Learning Algorithms". English. In: *Machine Learning* 38.3 (2000), pp. 257–286. ISSN: 0885-6125.

[22] Salvador García, Julián Luengo, and Francisco Herrera. "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining". In: *Knowledge-Based Systems* 98 (2016), pp. 1 –29. ISSN: 0950-7051.

DROP: the genuine IS algorithm

Algorithm 3: Decremental Reduction Optimization Procedure 1

Input: Training set $T = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$

Result: Instance set $S \subseteq T$

Let $S = T$

foreach *instance* $\mathbf{x} \in S$ **do**

  Find $\mathbf{x}.N_{1\dots k+1}$, the $k+1$ nearest neighbours of $\mathbf{x}$ in S

  Add $\mathbf{x}$ to each of its neighbours' lists of associates

end

foreach *instance* $\mathbf{x} \in S$ **do**

  Let `with` = # of associates of $\mathbf{x}$ classified correctly with $\mathbf{x}$ as a neighbour

  Let `without` = # of associates of $\mathbf{x}$ classified correctly without $\mathbf{x}$ as a neighbour

  if `without` $\geq$ `with` **then**

    Remove $\mathbf{x}$ from S

    foreach *associate* $\mathbf{a}$ *of* $\mathbf{x}$ **do**

      Remove $\mathbf{x}$ from $\mathbf{a}$'s list of nearest neighbours

      Find a new nearest neighbour for $\mathbf{a}$

      Add $\mathbf{a}$ to its new neighbour's list of associates

    end

    foreach *neighbour* $\mathbf{n}$ *of* $\mathbf{x}$ **do** Remove $\mathbf{x}$ from $\mathbf{n}$'s list of associates

  end

end

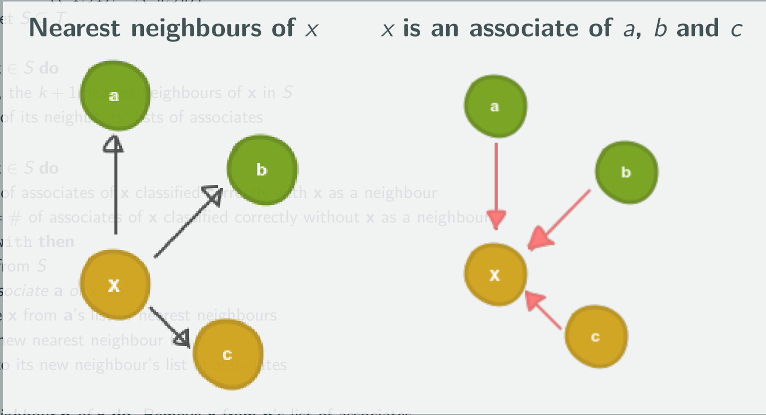
return S

DROP: the genuine IS algorithm

Algorithm 3: Decremental Reduction Optimization Procedure 1

Input: Training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$
Result: Instance set $S \subseteq T$

```
Let S = T
foreach instance x ∈ S do
    Find x.N_{1...k+1}, the k + 1 nearest neighbours of x in S
    Add x to each of its neighbours' lists of associates
end
foreach instance x ∈ S do
    Let with = # of associates of x classified correctly with x as a neighbour
    Let without = # of associates of x classified correctly without x as a neighbour
    if without ≥ with then
        Remove x from S
        foreach associate a of x
            Remove x from a's list of nearest neighbours
            Find a new nearest neighbour for a
            Add a to its new neighbour's list of associates
        end
        foreach neighbour n of x do  Remove x from n's list of associates
    end
end
return S
```



DROP: the genuine IS algorithm

Algorithm 3: Decremental Reduction Optimization Procedure 1

Input: Training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$

Result: Instance set $S \subseteq T$

Let $S = T$

foreach *instance* $\mathbf{x} \in S$ **do**

    Find $\mathbf{x}.N_{1 \ldots k+1}$, the $k+1$ nearest neighbours of $\mathbf{x}$ in S

    Add $\mathbf{x}$ to each of its neighbours' lists of associates

end

foreach *instance* $\mathbf{x} \in S$ **do**

    Let `with` $= \#$ of associates of $\mathbf{x}$ classified correctly with $\mathbf{x}$ as a neighbour

    Let `without` $= \#$ of associates of $\mathbf{x}$ classified correctly without $\mathbf{x}$ as a neighbour

    if `without` $\geq$ `with` **then**

        Remove $\mathbf{x}$ from S

        foreach *associate* $\mathbf{a}$ *of* $\mathbf{x}$ **do**

            Remove $\mathbf{x}$ from $\mathbf{a}$'s list of nearest neighbours

            Find a new nearest neighbour for $\mathbf{a}$

            Add $\mathbf{a}$ to its new neighbour's list of associates

        end

        foreach *neighbour* $\mathbf{n}$ *of* $\mathbf{x}$ **do** Remove $\mathbf{x}$ from $\mathbf{n}$'s list of associates

    end

end

return S

The cornerstone: `with` and `without`

The key is how the counters `with` and `without` are computed.

Two ideas:

- DROP using error accumulation.
- DROP using thresholding.

DROP using error accumulation

Algorithm 4: Computation of `eWith` and `eWithout`

...

foreach *instance* $\mathbf{x} \in S$ **do**

  Let `eWith` $= 0$

  Let `eWithout` $= 0$

  foreach *associate* $\mathbf{a}$ *of* $\mathbf{x}$ **do**

    Add $|Y(\mathbf{a}) - \texttt{Model}(\mathbf{a}.N, \mathbf{a})|$ to `eWith`

    Add $|Y(\mathbf{a}) - \texttt{Model}(\mathbf{a}.N \setminus \mathbf{x}, \mathbf{a})|$ to `eWithout`

  end

  if `eWithout` $\leq$ `eWith` **then**

    Remove $\mathbf{x}$ from S

    ...

  end

end

DROP using thresholding

Algorithm 5: Computation of `with` and `without`

...
foreach *instance* $\mathbf{x} \in S$ **do**
  Let `with` $= 0$
  Let `without` $= 0$
  foreach a *associate of* **x do**
    $\theta_D = \alpha_D \cdot \mathrm{std}(Y(\mathbf{a}.N))$
    if $|Y(\mathbf{a}) - \mathtt{Model}(\mathbf{a}.N, \mathbf{a})| \leq \theta_D$ **then**
      Add 1 to `with`
    end
    if $|Y(\mathbf{a}) - \mathtt{Model}(\mathbf{a}.N \backslash \mathbf{x}, \mathbf{a})| \leq \theta_D$ **then**
      Add 1 to `without`
    end
  end
  if `without` $\geq$ `with` **then**
    Remove **x** from S
    ...
  end
end

Experimental setup

Regressors: kNN, MLP, and REPTree.

DROP algorithms tested:

- Using error accumulation (DROPx-RE): DROP2-RE and DROP3-RE.
- Using thresholding (DROPx-RT): DROP2-RT and DROP3-RT.

The `Model` used inside DROP: kNN ($k = 9$).

DROP was compared against RegCNN and the regressor trained over the whole data set.

Experiments were made without and with noise: 10%, 20% and 30%.

Data sets

	Dataset	# attributes	# instances	Correlation coefficient		
				*k*NN	MLP	REPTree
1	MachineCPU	6	209	0.9335	0.9433	0.8127
2	Baseball	16	337	0.8291	0.7350	0.7775
3	DEE	6	365	0.9013	0.9061	0.8631
4	AutoMPG8	7	392	0.9276	0.9330	0.9133
5	AutoMPG6	5	392	0.9345	0.9277	0.9081
6	Ele-1	2	495	0.8321	0.8402	0.7969
7	Stock	9	950	0.9927	0.9864	0.9832
8	Laser	4	993	0.9725	0.9873	0.9527
9	Concrete	8	1 030	0.8296	0.9103	0.8978
10	Treasury	15	1 049	0.9974	0.9981	0.9955
11	Mortgage	15	1 049	0.9981	0.9995	0.9965
12	Ele-2	4	1 056	0.9904	0.9969	0.9950
13	Friedman	5	1 200	0.9425	0.9135	0.8495
14	Wizmir	9	1 461	0.9930	0.9967	0.9926
15	Wankara	9	1 609	0.9924	0.9964	0.9912
16	Plastic	2	1 650	0.8773	0.9024	0.8606
17	Quake	3	2 178	0.1074	0.0808	0.0699
18	ANACALT	7	4 052	0.9755	0.9890	0.9898
19	Abalone	8	4 177	0.7253	0.7516	0.6918
20	Delta-ail	5	7 129	0.8267	0.8314	0.8035
21	Compactiv	21	8 192	0.9860	0.9903	0.9839
22	Puma32h	32	8 192	0.4286	0.3200	0.9562
23	Delta-elv	6	9 517	0.7768	0.7913	0.7760
24	Ailerons	40	13 750	0.8966	0.8947	0.8728
25	Pole	26	14 998	0.9807	0.9491	0.9851
26	Elevators	18	16 599	0.8487	0.9499	0.8448
27	California	8	20 640	0.8438	0.8468	0.8612
28	House	16	22 784	0.6863	0.6736	0.6827
29	Mv	10	40 768	0.9853	0.9999	0.9995

Results: measures

Instance selection is a multi-objective problem.

For joining into a single measure:

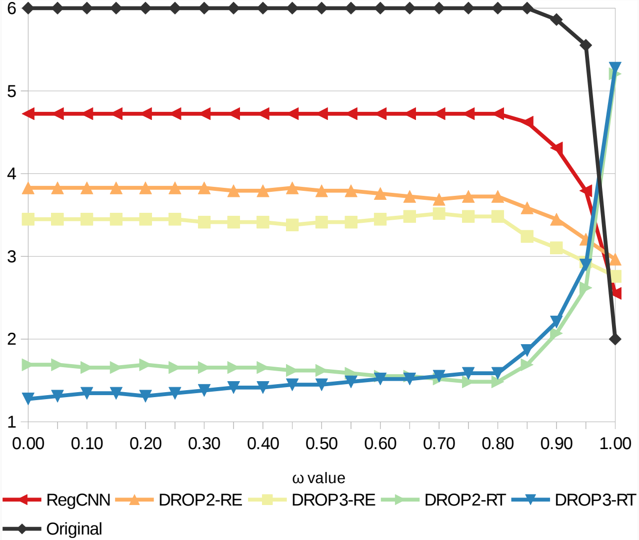
$$I_\omega = \omega \cdot \epsilon + (1 - \omega) \cdot m \tag{5}$$

For ϵ (error): $1 -$ *correlation coefficient* was used.

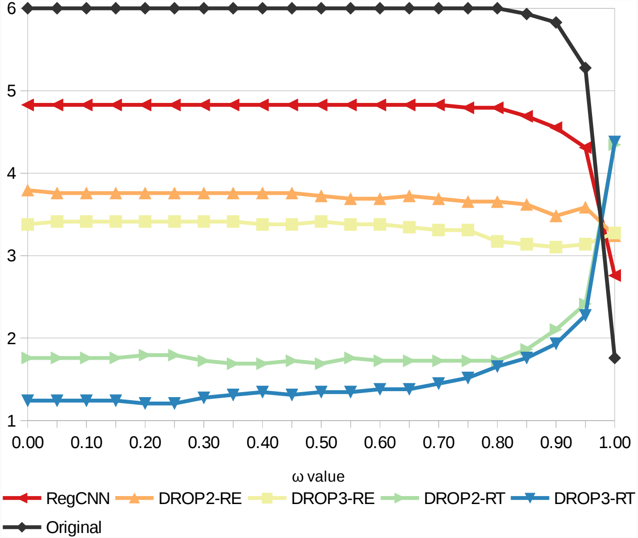
For m (retention ratio):

$$m = \frac{|\texttt{instances after selection}|}{|\texttt{instances before selection}|} \tag{6}$$

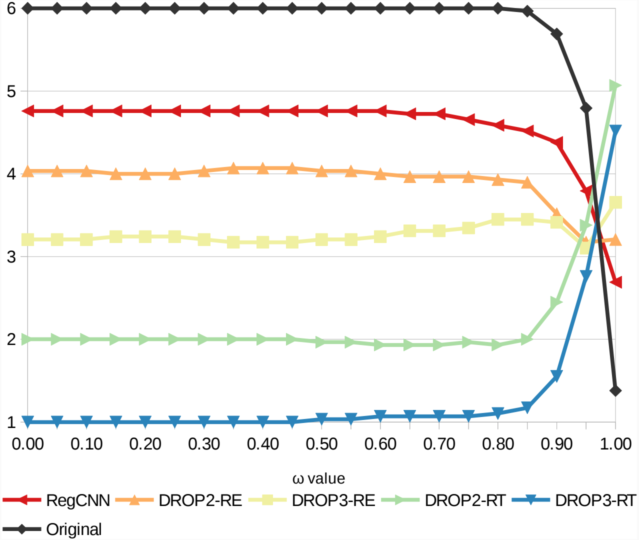
Results: *k*NN (no noise)



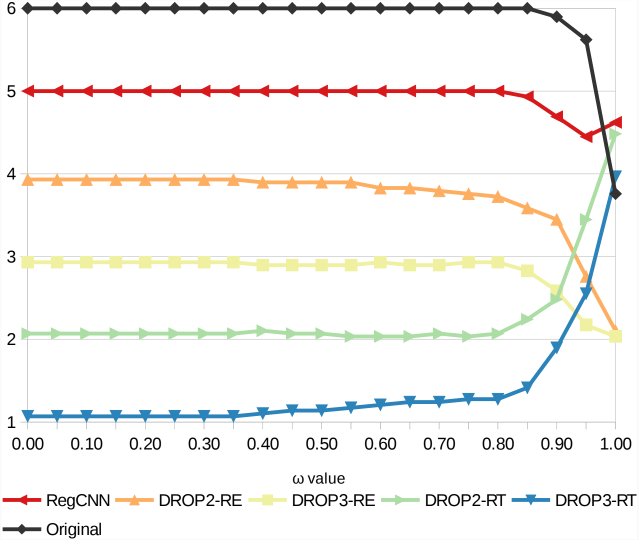
Results: MLP (no noise)



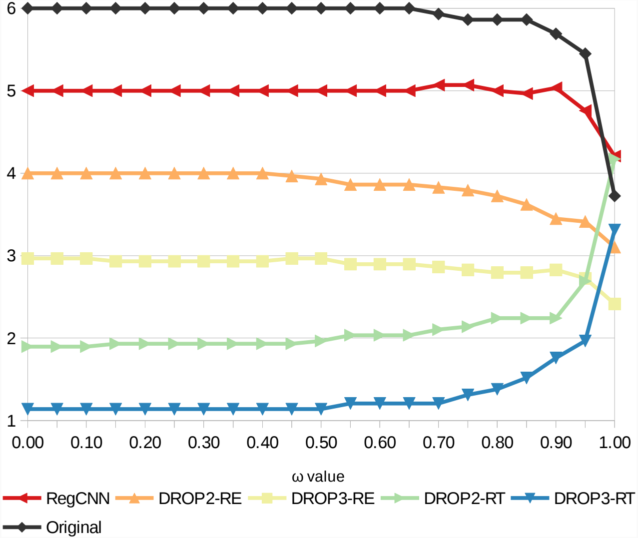
Results: REPTree (no noise)



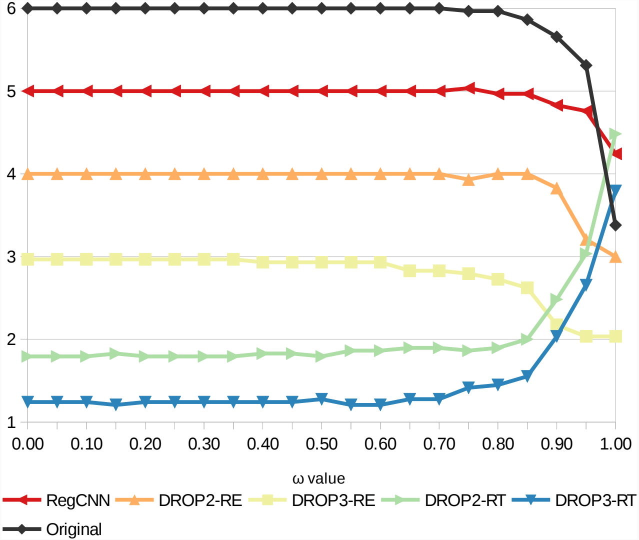
Results: kNN (10% noise)



Results: MLP (10% noise)



Results: REPTree (10% noise)



Results: compression

Average compression and Hochberg procedure.

IS algorithms	**Avg. rank**	**Avg. compression**
DROP3-RT	1.2931	0.532
DROP2-RT	1.7069	0.529
DROP2-RE	3.4483	0.366 ✖
DROP3-RE	3.8276	0.369 ✖
RegCNN	4.7241	0.213 ✖

Conclusions

More accuracy

DROPx-RE: DROP using error accumulation

More compression

DROPx-RT: DROP using thresholding

DROP3-Rx outperforms significantly DROP2-Rx: noise filter gives an advantage.

RegCNN is highly affected by noise: as the original CNN is.

7. Instance selection of linear complexity for big data

Locality-sensitive hasing (LSH)

Instance selection methods suffer from high computational complexity.

Hashing functions perform in linear time.

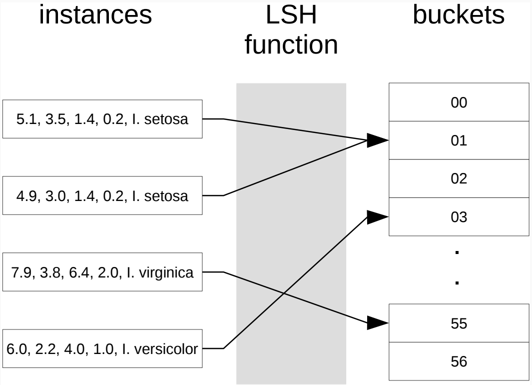
The idea

Use LSH functions for designing fast instance selection methods

LSH: how does it work?

Common hashing functions try to avoid collisions: assigning different buckets to similar elements.

LSH functions does not: they assign the same bucket to elements that are close in the input space.

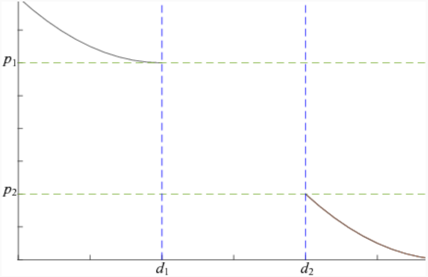


LSH functions (i)

Given a set of objects S and a distance measure D, a family of hash functions $\mathcal{H} = \{h : S \to U\}$ is said to be (d_1, d_2, p_1, p_2)-sensitive, if all functions of h in the family $\mathcal{H}$ follow:

- For all x, y in S, if $D(x, y) \leq d_1$, then the probability that $h(x) = h(y)$ is at least p_1.
- For all x, y in S, if $D(x, y) > d_2$, then the probability that $h(x) = h(y)$ is at most p_2.

It is possible to make the distances d_1 and d_2 can be as close as possible, but the cost will be that p_1 and p_2 are also closer. However, it is possible to combine families of hash functions that separate the probabilities p_1 and p_2 without modifying the distances d_1 and d_2.



LSH functions (ii)

The hash functions in the base family were obtained using the following equation[23].

$$h_{\vec{a},b}(\vec{x}) = \left\lfloor \frac{\vec{a} \cdot \vec{x} + b}{w} \right\rfloor \tag{7}$$

- $\vec{a}$ is a random vector (Gaussian distribution with mean 0 and standard deviation 1)
- b is a random real value from the interval $[0, w]$
- w is the width of each bucket in the hash table

This equation gives a $(w/2, 2w, 1/2, 1/3)$-sensitive family.

[23] Mayur Datar et al. "Locality-sensitive Hashing Scheme Based on P-stable Distributions". In: *Proceedings of the Twentieth Annual Symposium on Computational Geometry*. SCG '04. Brooklyn, New York, USA: ACM, 2004, pp. 253–262. ISBN: 1-58113-885-7.

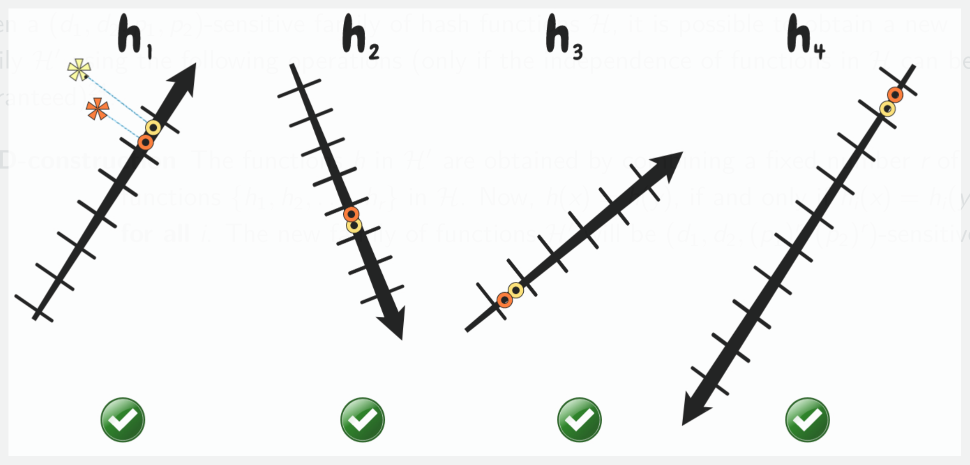
Given a (d_1, d_2, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$, it is possible to obtain a new family $\mathcal{H}'$ using the following operations (only if the independence of functions in $\mathcal{H}$ can be guaranteed)[24]:

AND-construction The functions h in $\mathcal{H}'$ are obtained by combining a fixed number r of functions $\{h_1, h_2, \ldots, h_r\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ **for all** i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.

[24] The AND-construction decreases the probabilities and the OR-construction increases them.

Given a (d_1, d_2, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$, it is possible to obtain a new family $\mathcal{H}'$ using the following operations (only if the independence of functions in $\mathcal{H}$ can be guaranteed)[24]:

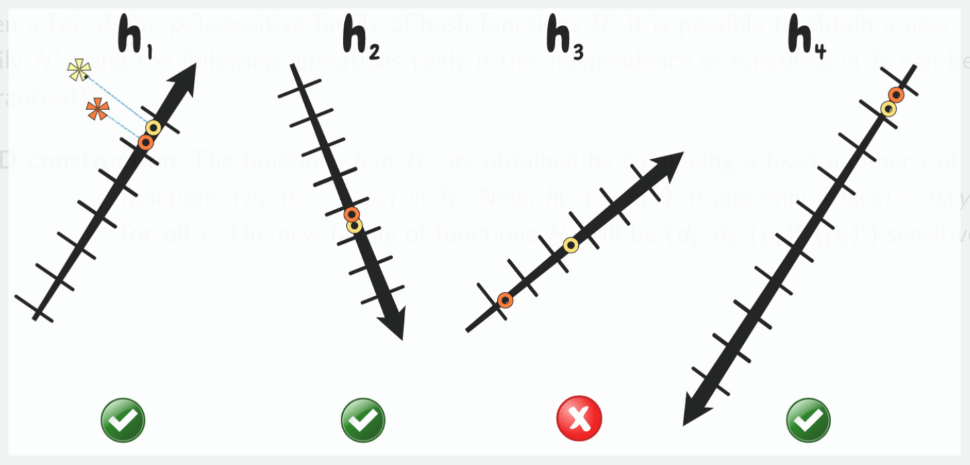
AND-construction: The functions h in $\mathcal{H}'$ are obtained by combining a fixed number r of functions $\{h_1, h_2, \ldots, h_r\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.



[24] The AND-construction decreases the probabilities and the OR-construction increases them.

Given a (d_1, d_2, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$, it is possible to obtain a new family $\mathcal{H}'$ using the following operations (only if the independence of functions in $\mathcal{H}$ can be guaranteed)[24]:

AND-construction: The functions h in $\mathcal{H}'$ are obtained by combining a fixed number r of functions $\{h_1, h_2, \ldots, h_r\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for all i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.



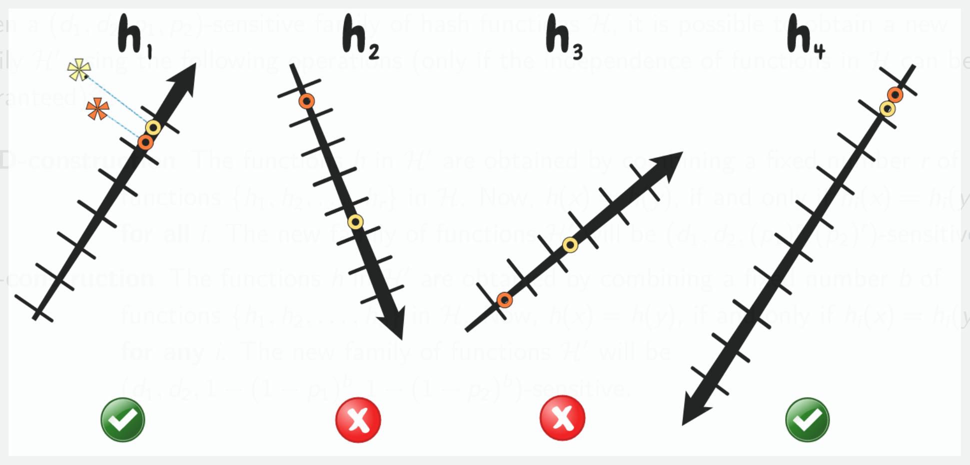
[24]The AND-construction decreases the probabilities and the OR-construction increases them.

Given a (d_1, d_2, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$, it is possible to obtain a new family $\mathcal{H}'$ using the following operations (only if the independence of functions in $\mathcal{H}$ can be guaranteed)[24]:

AND-construction The functions h in $\mathcal{H}'$ are obtained by combining a fixed number r of functions $\{h_1, h_2, \ldots, h_r\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ **for all** i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.

OR-construction The functions h in $\mathcal{H}'$ are obtained by combining a fixed number b of functions $\{h_1, h_2, \ldots, h_b\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ **for any** i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, 1 - (1 - p_1)^b, 1 - (1 - p_2)^b)$-sensitive.

[24] The AND-construction decreases the probabilities and the OR-construction increases them.



Given a (d_1, d_2, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$, it is possible to obtain a new family $\mathcal{H}'$ using the following operations (only if the independence of functions in $\mathcal{H}$ can be guaranteed):

AND-construction: The functions h in $\mathcal{H}'$ are obtained by combining a fixed number r of functions $\{h_1, h_2, \ldots, h_r\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ for all i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.

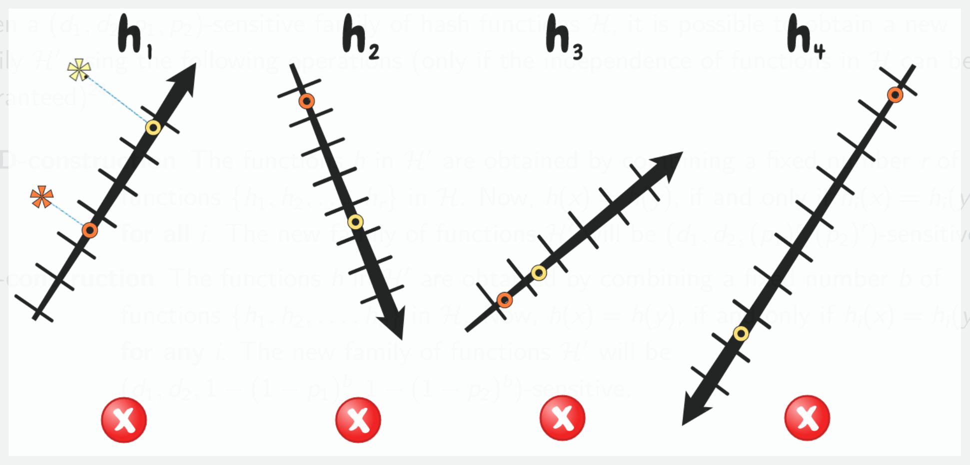
OR-construction: The functions h in $\mathcal{H}'$ are obtained by combining a fixed number b of functions $\{h_1, h_2, \ldots, h_b\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ for any i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive.

[24] The AND-construction decreases the probabilities and the OR-construction increases them.

Given a (d_1, d_2, p_1, p_2)-sensitive family of hash functions $\mathcal{H}$, it is possible to obtain a new family $\mathcal{H}'$ using the following operations (only if the independence of functions in $\mathcal{H}$ can be guaranteed)[24]

AND-construction The functions h in $\mathcal{H}'$ are obtained by combining a fixed number r of functions $\{h_1, h_2, \ldots, h_r\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ for all i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, (p_1)^r, (p_2)^r)$-sensitive.

OR-construction The functions h in $\mathcal{H}'$ are obtained by combining a fixed number b of functions $\{h_1, h_2, \ldots, h_b\}$ in $\mathcal{H}$. Now, $h(x) = h(y)$, if and only if $h_i(x) = h_i(y)$ for any i. The new family of functions $\mathcal{H}'$ will be $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-sensitive.



[24] The AND-construction decreases the probabilities and the OR-construction increases them.

LSH-IS-S: one pass

Algorithm 6: LSH-IS-S: Instance selection algorithm by hashing in one pass.

Input: A training set $X = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, set $\mathcal{G}$ of hash function families
Output: The set of selected instances $S \subseteq X$

```
S = ∅
foreach instance x ∈ X do
    foreach function family g ∈ G do
        u ← bucket assigned to x by family g
        if there is no other instance of the same class of x in u then
            Add x to S
            Add x to u
        end
    end
end
return S
```

LSH-IS-F: two passes

Algorithm 7: LSH-IS-F: Instance selection algorithm by hashing with two passes.

Input: A training set $X = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, set $\mathcal{G}$ of hash function families

Output: The set of selected instances $S \subseteq X$

$S = \varnothing$
foreach *instance* $\mathbf{x} \in X$ **do**
    foreach *function family* $g \in \mathcal{G}$ **do**
        $u \leftarrow$ bucket assigned to $\mathbf{x}$ by family g
        Add $\mathbf{x}$ to u
    end
end
foreach *function family* $g \in \mathcal{G}$ **do**
    foreach *bucket* u *of* g **do**
        foreach *class* y *with some instance in* u **do**
            $I_y \leftarrow$ all instances of class y in u
            if $|I_y| > 1$ **then**
                Add to S one random instance of I_y
            end
        end
    end
end
return S

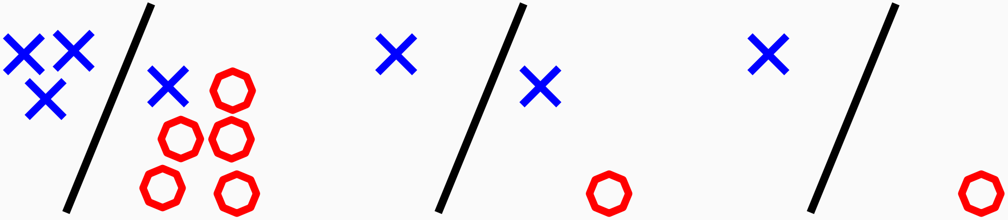
How do they work? An example

Two buckets are identified by LSH and the line shows the boundary:

(a) Initial instances.

(b) Instances selected by LSH-IS-S.

(c) Instances selected by LSH-IS-F.

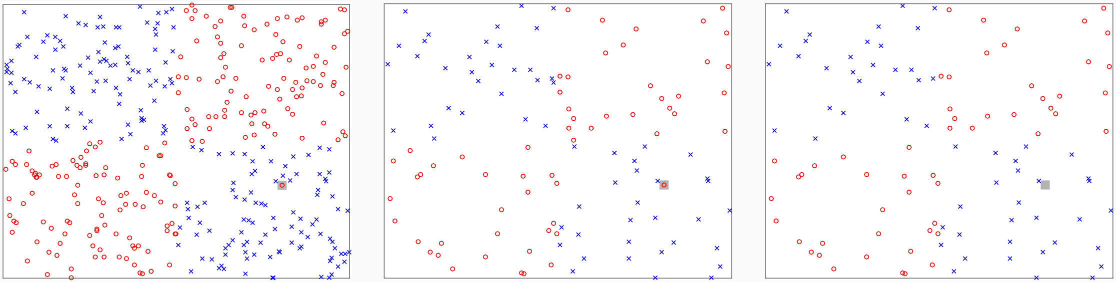


(a) (b) (c)

How do they work? XOR example

Example with XOR data set:

(a) Original data set, an outlier is highlighted in gray.
(b) LSH-IS-S maintains the outlier.
(c) LSH-IS-F removes the outlier.



(a) (b) (c)

Data sets: small and medium size

	Data sets	# attributes		# instances	# classes	Accuracy	
		Continuous	Nominal			1NN	J48
1	German	7	13	1 000	2	72.90	71.80
2	Flare	0	11	1 066	6	73.26	73.55
3	Contraceptive	9	0	1 473	3	42.97	53.22
4	Yeast	8	0	1 484	10	52.22	56.74
5	Wine-quality-red	11	0	1 599	11	64.85	62.04
6	Car	0	6	1 728	4	93.52	92.36
7	Titanic	3	0	2 201	2	79.06	79.06
8	Segment	19	0	2 310	7	97.23	96.62
9	Splice	0	60	3 190	3	74.86	94.17
10	Chess	0	35	3 196	2	72.12	81.85
11	Abalone	7	1	4 174	29	19.84	20.72
12	Spam	0	57	4 597	2	91.04	92.97
13	Wine-quality-white	11	0	4 898	11	65.40	58.23
14	Banana	2	0	5 300	2	87.21	89.04
15	Phoneme	5	0	5 404	2	90.19	86.42
16	Page-blocks	10	0	5 472	5	95.91	97.09
17	Texture	40	0	5 500	11	99.04	93.13
18	Optdigits	63	0	5 620	10	98.61	90.69
19	Mushroom	0	22	5 644	2	100.00	100.00
20	Satimage	37	0	6 435	7	90.18	86.28
21	Marketing	13	0	6 876	10	28.74	31.06
22	Thyroid	21	0	7 200	3	92.35	99.71
23	Ring	20	0	7 400	2	75.11	90.95
24	Twonorm	20	0	7 400	2	94.81	85.12
25	Coil 2000	85	0	9 822	2	90.62	93.95
26	Penbased	16	0	10 992	10	99.39	96.53
27	Nursery	0	8	12 960	5	98.13	97.13
28	Magic	10	0	19 020	2	80.95	85.01
29	Letter	16	0	20 000	27	96.04	87.98
30	KR vs. K	0	6	28 058	18	73.05	56.58

Experimental setup: classifiers and IS methods

Classifiers used: 1NN and J48.

The proposed methods were compared against:

- CNN, MSS[25], HMN-EI[26], and LSBo.
- ICF[27] and DROP3: $k = 3$.
- PSC[28]: $num\ clusters = 6r$.

[25] Ricardo Barandela, Francesc J. Ferri, and José Salvador Sánchez. "Decision boundary preserving prototype selection for nearest neighbor classification". In: *IJPRAI* 19.6 (2005), pp. 787–806.
[26] Elena Marchiori. "Hit Miss Networks with Applications to Instance Selection". In: *J. Mach. Learn. Res.* 9 (June 2008), pp. 997–1017. ISSN: 1532-4435.
[27] Henry Brighton and Chris Mellish. "Advances in Instance Selection for Instance-Based Learning Algorithms". English. In: *Data Mining and Knowledge Discovery* 6.2 (2002), pp. 153–172. ISSN: 1384-5810.
[28] J.Arturo Olvera-López, J.Ariel Carrasco-Ochoa, and J.Francisco Martínez-Trinidad. "A new fast prototype selection method based on clustering". In: *Pattern Analysis and Applications* 13.2 (2009), pp. 131–141. ISSN: 1433-755X.

Results: accuracy

Average ranks over accuracy: 1NN classifier (left) and J48 (right).

Algorithm	Ranking	p Hoch.
HMN-EI	2.92	
LSBo	3.85	0.1869
LSH-IS-F	4.45	0.0602
MSS	4.58	0.0553
LSH-IS-S	4.98	0.0139
DROP3	5.03	0.0138
CNN	5.17	0.0088
ICF	5.75	0.0004
PSC	8.27	0.0000

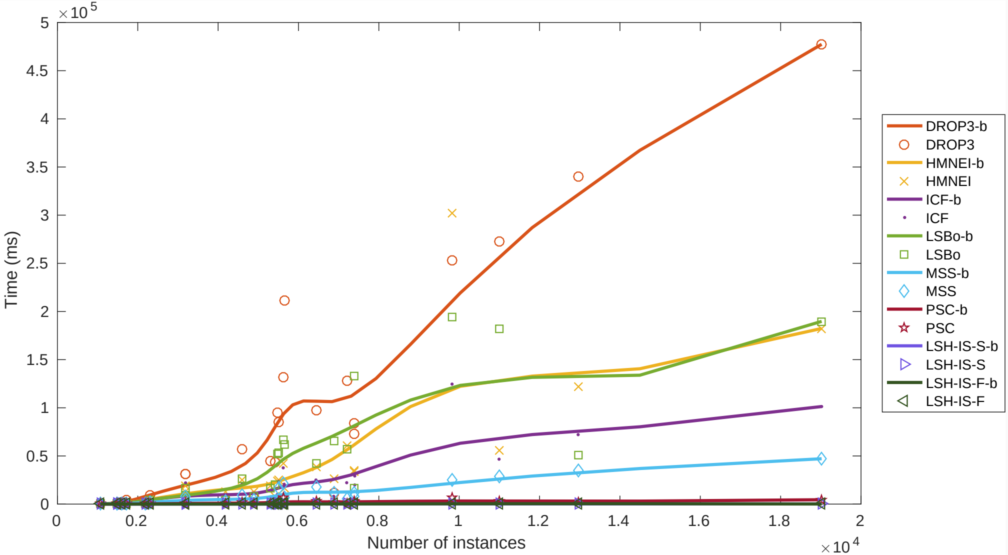
Algorithm	Ranking	p Hoch.
LSH-IS-F	3.63	
LSH-IS-S	3.88	0.7237
HMN-EI	4.10	0.7237
LSBo	4.57	0.5606
MSS	5.03	0.1909
CNN	5.28	0.0981
ICF	5.67	0.0242
DROP3	5.90	0.0094
PSC	6.93	0.0000

Results: compression

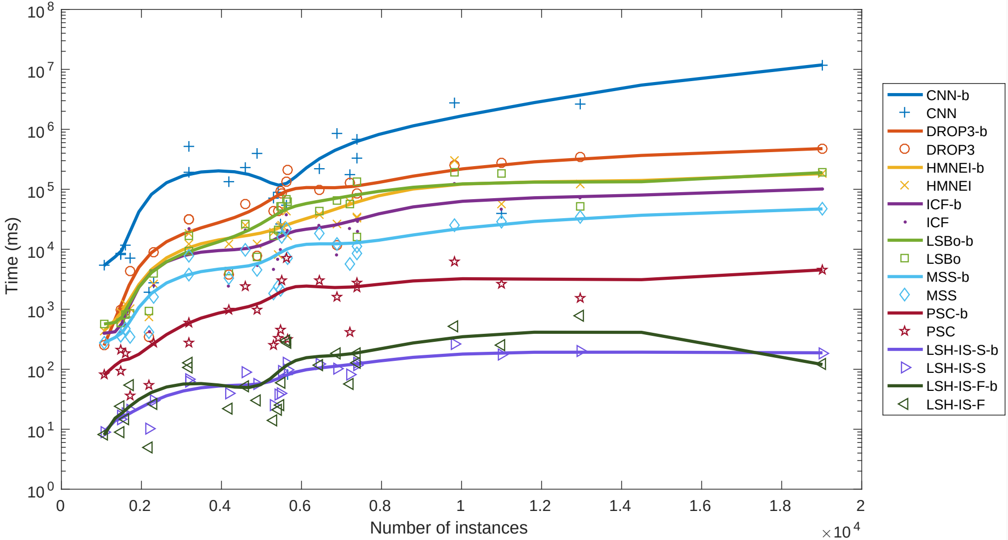
Average ranks and Hochberg procedure over storage reduction, and average reduction rate.

Algorithm	Ranking	p Hoch.	Reduction rate
DROP3	1.67		0.896
ICF	3.10	0.0427	0.813
LSBo	3.70	0.0081	0.737
PSC	4.70	0.0001	0.762
CNN	5.43	0.0000	0.658
MSS	6.00	0.0000	0.665
HMN-EI	6.10	0.0000	0.577
LSH-IS-F	6.62	0.0000	0.455
LSH-IS-S	7.68	0.0000	0.405

Results: filtering time



Results: filtering time (Log. scale)



Data sets: big and huge

	Data sets	# attributes		# instances	# classes	Accuracy
		Continuous	Nominal			
31	Census	7	30	299 285	2	92.70
32	KDDCup99	33	7	494 021	23	99.95
33	CovType	54	0	581 012	7	94.48
34	KDDCup991M	33	7	1 000 000	23	99.98
35	Poker	5	5	1 025 010	10	50.61

Experimental setup: classifier and IS methods

Classifier used: 1NN.

The proposed methods were compared against: Democratic Instance Selection[29]:

- RNN.
- ICF and DROP3: $k = 3$.

[29] César García-Osorio, Aida de Haro-García, and Nicolás García-Pedrajas. "Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts". In: *Artificial Intelligence* 174.5-6 (2010), pp. 410–441. ISSN: 0004-3702.

Results: accuracy and compression

Average ranks: accuracy (left) and compression (right).

Algorithm	Ranking
LSH-IS-F	2.0
DIS.RNN	2.2
LSH-IS-S	2.6
DIS.DROP3	3.6
DIS.ICF	4.6

Algorithm	Ranking
DIS.RNN	1.8
DIS.DROP3	2.4
LSH-IS-F	3.2
DIS.ICF	3.4
LSH-IS-S	4.2

Conclusions

- Linear complexity: $\mathcal{O}(n)$
- One of the methods does not need that the data fits in memory (*on-the-fly*).
- Competitive results as instance selection for medium and huge data sets.

8. Conclusions

Conclusions: regression

The lack of instance selection methods for regression sparked my initial interest in this task.

We designed and tested three ideas of instance selection for regression:

Conclusions: regression

The lack of instance selection methods for regression sparked my initial interest in this task.

We designed and tested three ideas of instance selection for regression:

- Discretization: to transform the continuous output variable into discrete counterparts $\Longrightarrow$ good performance on noise identification.

Conclusions: regression

The lack of instance selection methods for regression sparked my initial interest in this task.

We designed and tested three ideas of instance selection for regression:

- Discretization: to transform the continuous output variable into discrete counterparts $\Longrightarrow$ good performance on noise identification.
- Combination of several instance selection methods for regression was evaluated: it upholds the 'ensemble' hypothesis.

Conclusions: regression

The lack of instance selection methods for regression sparked my initial interest in this task.

We designed and tested three ideas of instance selection for regression:

- Discretization: to transform the continuous output variable into discrete counterparts $\Longrightarrow$ good performance on noise identification.
- Combination of several instance selection methods for regression was evaluated: it upshelds the 'ensemble' hypothesis.
- Adaptation of DROP to regression: it mimics the benefits of DROP for classification.

Conclusions: classification

High computational complexity of instance selection methods for classification.

We faced the problem from two points of view:

Conclusions: classification

High computational complexity of instance selection methods for classification.

We faced the problem from two points of view:

- Locality-Sensitive Hashing: linear complexity in relation to the number of instances.

Conclusions: classification

High computational complexity of instance selection methods for classification.

We faced the problem from two points of view:

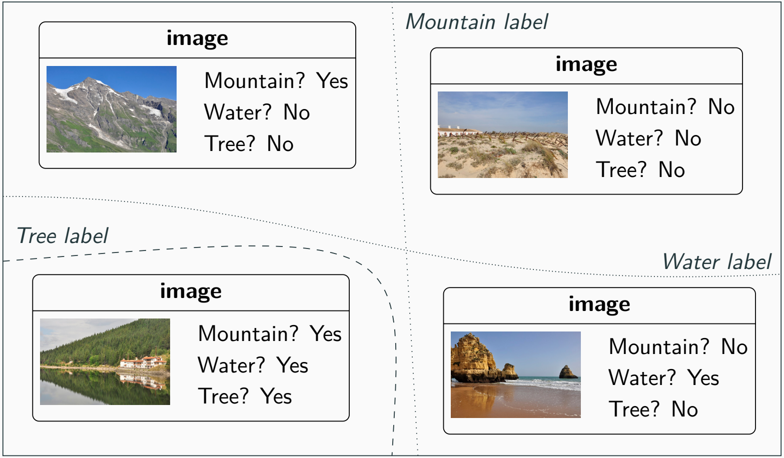
- Locality-Sensitive Hashing: linear complexity in relation to the number of instances.
- Democratic Instance Selection: we designed and implemented a parallel version of DIS method by following the MapReduce model[30].

[30] Álvar Arnaiz-González et al. "MR-DIS: Democratic Instance Selection for Big Data by MapReduce". In: *Progress in Artificial Intelligence* 6.3 (2017), pp. 211–219. ISSN: 2192-6360.

9. Current research

Current research: multi-label (i)

There is a relatively new topic where the usefulness of instance selection is starting to become apparent.



Current research: multi-label (ii)

The work on this topic has already borne fruit, as a result, the following couple of papers have recently been published:

- Local sets for multi-label instance selection[31].
- Study of data transformation techniques for adapting single-label prototype selection algorithms to multi-label learning[32].

---

[31] Álvar Arnaiz-González et al. "Local sets for multi-label instance selection". In: *Applied Soft Computing* (in press) (2018). ISSN: 1568-4946.

[32] Álvar Arnaiz-González et al. "Study of data transformation techniques for adapting single-label prototype selection algorithms to multi-label learning". In: *Expert Systems with Applications* (in press) (2018). ISSN: 0957-4174.

Current research: multi-target regression

A topic closely related to multi-label is multi-target regression.

Each instance in multi-target data sets has a group of numeric output values, not a set of labels.

There are many real-world problems to which multi-target regression can be applied: environmental sciences, bio-informatics, medicine...

There are not any instance selection algorithm for multi-target regression in the literature.

Thanks!

Questions

Álvar Arnaiz González

e-mail: alvarag@ubu.es

GitHub: github.com/alvarag/

ORCID: orcid.org/0000-0001-6965-0237

dblp: dblp.uni-trier.de/pers/hd/a/Arnaiz=Gonz=aacute=lez:=Aacute=lvar

Summary

Get the source of this theme and the demo presentation from

`github.com/matze/mtheme`

Appendix

10. Discretization

Discretization algorithm (i)

An unsupervised filter incorporated in Weka was used for discretization.

The equal-width option was selected, so all bins in which the target attribute was split had the same size.

The number of bins is selected from one to ten by the Weka filter using leave-one-out cross-validation to select the best way of separating the numerical output variable, i.e. the one that maximizes the entropy.

Discretization algorithm (ii)

Algorithm 8: Equal-width binning using leave-one-out estimated entropy.

Input: Training set $T = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$

Maximum number of bins b

Output: Discretized set $D = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$

$bestEntropy \leftarrow MAX\_VALUE$

for $i = 1 \ldots b$ **do**

  $entropy = \texttt{LOUEstimatedEntropy}(T, i)$

  if $entropy < bestEntropy$ **then**

    $bestEntropy \leftarrow entropy$

    $bestNumBins \leftarrow i$

  end

end

$cutPoints = \texttt{CalcCutPoints}(T, i)$

$D = \texttt{DiscretizeClass}(T, cutPoints)$

return D

RegENN

Algorithm 9: RegENN: Edited Nearest Neighbour for regression using a threshold

Input: Training set $T = \{(\mathbf{x}_1, y_1), \ldots (\mathbf{x}_n, y_n)\}$, parameter α to control how the threshold is calculated from the standard deviation

Output: Instance set $P \subseteq T$

for $i = 1 \ldots n$ **do**
    $\bar{Y}(\mathbf{x}_i) = \texttt{Model}(T \backslash \mathbf{x}_i, \mathbf{x}_i)$
    $S = \texttt{kNN}(T, \mathbf{x}_i)$
    $\theta = \alpha \cdot std\big(Y(X_S)\big)$
    if $|Y(\mathbf{x}_i) - \bar{Y}(\mathbf{x}_i)| > \theta$ **then**
        $T \leftarrow T \backslash \mathbf{x}_i$
    end
end
$P \leftarrow T$
return P

Class noise

The noise configurations were tested at 10, 20, 30 and 40% adding or subtracting a random value to the target attribute.

Mutual Information IS

Algorithm 10: Algorithm based on mutual information

Input: Training set $\{X, Y\} = \{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_n, y_n)\}$, the number k of neighbours

Output: Edited set $S \subseteq \{X, Y\}$

$S = \varnothing$
for $i = 1 \dots n$ **do**
    Calculate $\mathtt{NN}[\mathbf{x}_i, j]$, the k nearest neighbours ($j = 1 \dots k$) in the input space
end
for $i = 1 \dots n$ **do**
    Calculate the value of mutual information $I(X, Y)_i$ when $\mathbf{x}_i$ is eliminated from X
end
Normalize $I(X, Y)_i$ in $[0, 1]$
for $i = 1 \dots n$ **do**
    $Cdiff = 0$
    for $j = 1 \dots k$ **do**
        $diff = I(X, Y)_i - I(X, Y)_{NN[\mathbf{x}_i, j]}$
        if $diff > \alpha$ **then** $Cdiff = Cdiff + 1$
    end
    if $Cdiff < k$ **then** add $(\mathbf{x}_i, y_i)$ to S
end
return S

11. Fusion

T-CNN

Algorithm 11: T-CNN: Condensed Nearest Neighbour for regression using a threshold

Input: Training set $T = \{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_n, y_n)\}$, parameter α to control how the threshold is calculated from the standard deviation

Output: Instance set $P \subseteq T$

$P = \varnothing$

$P \leftarrow P \cup \mathbf{x}_1$

for $i = 2 ... n$ **do**

  $\bar{Y}(\mathbf{x}_i) = \texttt{Model}(P, \mathbf{x}_i)$

  $S = \texttt{kNN}(T, \mathbf{x}_i)$

  $\theta = \alpha \cdot std\big(Y(X_S)\big)$

  if $|Y(\mathbf{x}_i) - \bar{Y}(\mathbf{x}_i)| > \theta$ **then**

    $P \leftarrow P \cup \mathbf{x}_i$

    $T \leftarrow T \backslash \mathbf{x}_i$

  end

end

return P

Experimental setup

Bagging parameters:

- The bagging ensemble was set to 30 members.
- Each subset was created by randomly drawing instances without replacement.
- The number of instances in the subset was 80% of the original.

Other parameters:

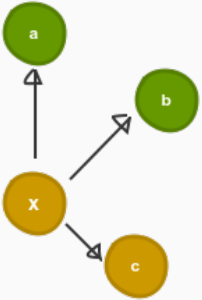
- Number of k-nearest neighbours used by kNN: from 1 to 13 in steps of 2.
- Number of k-nearest neighbours used by instance selection algorithms: from 1 to 13 in steps of 2.
- Threshold controlled by α: from 0.1 to 1 in steps of 0.1.
- Maximum number of bins D: from 5 to 15 in steps of 1.
- Percentage of votes to select an instance z: from 0.1 to 0.9 in steps of 0.1.

12. DROP for regression

Nearest neighbours and associate concepts

The nearest neighbours of x are a, b, and c instances.

$$\mathrm{NN}(x) = \{a, b, c\}$$

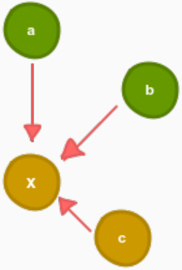


Therefore, a, b, and c have x as associate.

$$x \in \mathrm{associate}(a)$$

$$x \in \mathrm{associate}(b)$$

$$x \in \mathrm{associate}(c)$$



Differences between DROP algorithms: (i)

The differences between the variants of DROP methods are as follows:

- DROP1 eliminates an instance **p** of S, if its associates in S are correctly classified without **p**, that is, if the elimination of **p** does not affect the classification results.
- The DROP2 removes an instance[33]; **p** of S if the associates that **p** has in the original set, T, are correctly classified without **p**. Before starting the selection, it sorts the instances in descending order from their distance to their *nearest enemy*. In this way, instances are processed in an order that is the reverse of its distance to the class boundary, the furthest instance is processed first, then the second furthest, and so on.

[33] The DROP2, DROP3, DROP4 and DROP5 algorithms verify the effect that causes the removal of an instance on the original sample.

Differences between DROP algorithms: (ii)

The differences between the variants of DROP methods are as follows:

- The DROP3 algorithm, applies a noise filter before starting. The filtering state removes all instances that are not correctly classified by their k nearest neighbours.
- DROP4 is identical to DROP3 but applies a slightly different noise filter, involving the removal of an instance only if it is misclassified by its k nearest neighbours and its removal does not mean that another instance is badly classified. This avoids the removal of too many instances in the filtering stage.
- The DROP5 algorithm is similar to DROP2, but it begins to analyse the instances that are found close to its nearest enemy (those on the class boundary).

Experimental setup: regressors

The regressors used were:

- nearest neighbours (with $k = 8$).
- multilayer perceptron (trained with backpropagation with the following parameters: learning rate $= 0.3$, momentum $= 0.2$, number of hidden neurons $= \frac{\#attr+1}{2}$).
- REPTrees (with Weka default parameters).

Class noise

In the experiments that were carried out, the noise was introduced by exchanging the output values of two randomly selected instances.

This way, the distribution of the sample, both for the input variables and for the output variables was not modified.

Experimental setup: DROPx-Rx

Value of α_E which has achieved lower errors for the different regressors and noise levels.

Noise	**IS algorithm**	***k*NN**	**MLP**	**REPTree**
0 %	DROP3-RE	5	5	3
	DROP2-RT	4	2	1
	DROP3-RT	4	5	2
10 %	DROP3-RE	3	2	2
	DROP2-RT	3	5	2
	DROP3-RT	3	2	3
20 %	DROP3-RE	2	2	2
	DROP2-RT	3	1	3
	DROP3-RT	4	2	1
30 %	DROP3-RE	2	1	1
	DROP2-RT	1	3	5
	DROP3-RT	4	1	1

Experimental setup: RegCNN

It is influenced by two parameters α and *k*.

With the aim to use reasonable values for these parameters, we first launched several experiments with different values: 0.25, 0.5, 0.75 and 1 for α; and 3, 5, 7 and 9 for *k*.

The best results, for all regressors, were achieved using $\alpha = 0.25$.

The optimal values for *k* depended on the regressor, as the table shows:

Noise	***k*NN**	**MLP**	**REPTree**
0 %	9	9	9
10 %	3	7	3
20 %	9	7	9
30 %	7	7	7

13. LSH-IS

Complexity

Summary of state-of-the-art IS methods (taxonomy from[34]; computational complexity from[35] and authors' papers).

Strategy	Direction	Algorithm	Complexity	Year
Condensation	Incremental	CNN	$\mathcal{O}(n^3)$	1968
	Incremental	PSC	$\mathcal{O}(n \log n)$	2010
	Decremental	RNN	$\mathcal{O}(n^3)$	1972
	Decremental	MSS	$\mathcal{O}(n^2)$	2002
Hybrid	Decremental	DROP1-5	$\mathcal{O}(n^3)$	2000
	Batch	ICF	$\mathcal{O}(n^2)$	2002
	Batch	HMN-EI	$\mathcal{O}(n^2)$	2008
	Batch	LSBo	$\mathcal{O}(n^2)$	2015

[34] S. Garcia et al. "Prototype Selection for Nearest Neighbor Classification: Taxonomy and Empirical Study". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.3 (2012), pp. 417–435. ISSN: 0162-8828.

[35] Norbert Jankowski and Marek Grochowski. "Comparison of Instances Seletion Algorithms I. Algorithms Survey". English. In: *Artificial Intelligence and Soft Computing - ICAISC 2004*. Ed. by Leszek Rutkowski et al. Vol. 3070. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, pp. 598–603. ISBN: 978-3-540-22123-4.

LSH functions (i)

The reason for $w/2$ and $1/2$ is: if the distance d between two points is exactly $w/2$ (half the width of the buckets) the smallest probability for the two points falling in the same segment would happen for $\theta = 0$, and in this case the probability would be 0.5, since d is exactly $w/2$. For angles greater than 0, this probability will be even higher; in fact, it will be 1 for $\theta = 90$. And for shorter distances than $w/2$, the probability will equally increase. So the lower boundary for this probability is $1/2$.

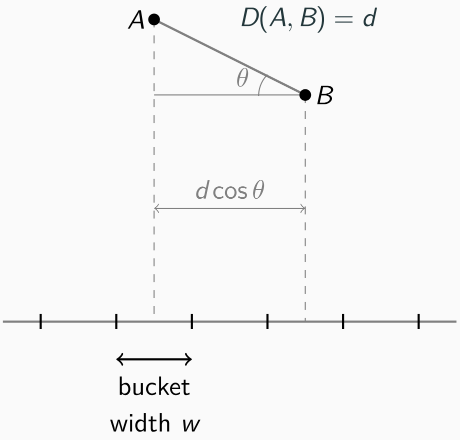


Figure 4: Two points (A, B) at distance $d \gg w$ have a small chance of being hashed to the same bucket.

LSH functions (ii)

The reason for $2w$ and $1/3$ is: if the distance d is exactly $2w$ (twice the width of the bucket), the only chance for both points to fall in the same bucket is that their distances, once projected in the segment, are lower than w, what means that $\cos\theta$ must be lower than 0.5, since the projected distance is $d\cos\theta$ and d is exactly $2w$. For θ in the interval 0 to 60, $\cos\theta$ is greater than 0.5, so the only chance of $\cos\theta$ being lower than 0.5 is that θ is in the interval $[60, 90]$, and the chance of that happening is at most $1/3$. For distances greater than $2w$, the probabilities are even lower. So the upper boundary of this probability is $1/3$.

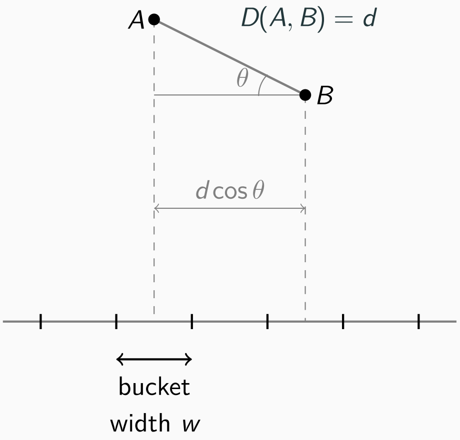


Figure 5: Two points (A, B) at distance $d \gg w$ have a small chance of being hashed to the same bucket.

LSH configuration: AND/OR constructions

- LSH-IS-S: the best configuration is one that uses OR-constructions of 6 functions obtained using an AND-construction on 10 functions of the base family.
- LSH-IS-F: the best results were obtained using OR-constructions of 5 functions obtained by combining by AND-construction 10 functions of the base family.

	1	2	3	4	5	6
1	58.015	56.030	54.030	53.091	52.197	51.682
2	54.000	49.939	47.742	46.121	44.667	43.348
3	49.970	45.152	40.303	41.091	35.333	33.848
4	43.864	37.712	32.106	32.167	30.242	29.727
5	38.030	30.742	28.636	27.545	24.848	24.197
6	36.242	27.424	25.924	24.273	21.652	20.727
7	32.530	23.652	21.167	20.167	17.409	15.833
8	29.682	20.636	18.636	16.439	14.288	13.288
9	26.879	18.303	16.091	14.788	12.818	12.333
10	24.182	17.136	14.712	13.212	11.667	11.530

	1	2	3	4	5	6
1	58.348	56.606	54.727	53.697	52.667	52.091
2	54.045	50.409	47.697	46.227	45.197	43.485
3	48.409	43.030	42.364	41.879	39.333	38.136
4	43.303	38.303	37.212	35.742	32.091	30.561
5	39.773	30.682	30.379	29.273	25.045	23.909
6	35.136	25.833	24.485	24.212	21.485	20.273
7	31.773	20.879	19.576	18.985	16.985	15.924
8	27.439	18.955	16.773	16.121	15.015	13.894
9	25.621	17.061	14.894	14.348	12.318	12.076
10	20.697	15.561	13.667	12.864	11.258	11.273