INSTANCE BASED LEARNING

Based on “Instance-based Learning Algorithms” by Aha, Kibler and Albert, Machine Learning, 6, 37-66, 1991

Cited by more than 3000

3-4-2013

Introduction

- The k-NN classification algorithm looks at k nearest neighbors of a new instance to decide which class the new instance should belong to.
- When k=1, we have the nearest neighbor algorithm.
- k-NN classification is incremental.
- k-NN classification does not really have a training phase; all instances are stored. Training may involve some kind of indexing to find neighbors quickly.
- During use or testing, k-NN classification algorithm has to find k nearest neighbors of a new instance. This is time consuming if we do exhaustive comparsion.
- In addition, storing all the instances in memory, necessary to compute k nearest neighbors, takes a lot of space also.

Instance-based Learning Algorithms

- Instance-based learning (IBL) are an extension of nearest neighbor or k-NN classification algorithms.
- IBL algorithms do not maintain a set of abstractions of model created from the instances.
- The k-NN, algorithms have large space requirement.
- Aha et al. (1991) discuss how the storage requirement can be reduced significantly, with only a minor loss in learning rate and accuracy.
- They also extend it with a significance test to work with noisy instances, since a lot of real-life datasets have training instances and k-NN algorithms do not work well with noise.

Why are k-NN type algorithms good?

- These are supervised learning algorithms where instances are used (incrementally) to classify objects.
- Cost for updating object instances is low.
- Learning rate or speed of learning (i.e., training) is fast.
- It is possible to extend the algorithms to obtain concept descriptions.

Why are k-NN or IBL algorithms bad?

- They are computationally expensive since they save all training instances.
- They are not good with noisy attribute values.
- They are not good with irrelevant attributes.
- The performance depends on the choice of the similarity function to compute distance between two instances.
- There is no simple or natural way to work with nominal-valued attributes or missing attributes.
- They do not tell us much about how the data is structured.

Instance-based Learning (IBL)

- IBL algorithms are supervised learning algorithms or they learn from labeled examples.
- IBL algorithms can be used incrementally, where the input is a sequence of instances.
- Each instance is described by *n* attribute-value pairs.
- One attribute is a category attribute.
- We assume that there is exactly one category attribute for each instance, and the categories are disjoint, although IBL algorithms can be extended to learn multiple, possibly overlapping categories simultaneously.

IBL Methodology and Framework

- *Concept Description (CD)*: This is the primary output of an IBL algorithm. It is some kind of a function that maps instances to categories, i.e., given an instance, it tells us which class it belongs to.
- *Stored Instances*: A CD includes a set of stored instances and possibly some information concerning their past performance during classification (e.g., their number of correct and incorrect classification predictions).
- The set of stored instances in a CD can change after each instance is processed.
- The CD for a concept is determined by how the IBL algorithm has performed on examples seen so far.

IBL Methodology and Framework…

- The framework requires three components: Similarity function, Classification function, and Concept Description Updater.
- *Similarity Function*: This computes similarity between a training instance the instances in the CD at a certain point in time. Similarities are numeric-valued.
- *Classification Function*: Given a new instance, it gives us the classification for that instance, based on the values coming from the similarity function, instances in the CD and performance of these instances in classification so far.
- *Concept Description Updater*: Maintains record of classification performance and decides which instances to include in the CD. Inputs: the new test instance, similarity values, classification results, and current CD. Outputs: A modified CD.

IBL Methodology and Framework…

- Learning Bias: IBL algorithms assume that *similar instances have similar classification*.
- In other words, IBL algorithms classify instances based on the classification of their most similar neighbors.
- IBL algorithms assume that, without prior knowledge, attributes have equal relevance for classification, although this can be changed by using weights.
- IBL algorithms do not construct abstractions such as decision trees or rules at the fundamental level, although such constructs can be obtained if necessary.
- IBL algorithms perform relatively little work during presentation time or training.
- The workload is higher when presented with new instances for classification.

The 3 Algorithms

- The paper presents three algorithms: IB1, IB2 and IB3
- IB1 is the simplest algorithm.
- IB2 improves IB1 so that the storage requirement is lowered.
- IB3 improves IB2 further to handle noisy instances.

IB1 Algorithm

$CD \leftarrow \emptyset$

for each $x \in$ Training Set **do**

1. **for each** $y \in CD$ **do**
   Sim[y] $\leftarrow$ Similarity(x, y)
2. $y_{max} \leftarrow$ some $y \in CD$ with maximal Sim[y]
3. **if** class(x) = class(y_{max})
   then classification $\leftarrow$ **correct**
   else classification $\leftarrow$ **incorrect**
4. $CD \leftarrow CD \cup \{x\}$

$$\text{Similarity}(x, y) = -\sqrt{\sum_{i=1}^{n} f(x_i, y_i)}$$

Given two instances *x* and *y*, the similarity is computed usign the function given above. i is the index to an attribute. For numeric attributes, use Euclidean distance. For non-numeric attributes is 1 if they are equal, 0 otherwise.
CD = Concept Description

IB1 Algorithm…

- The IB1 Algorithm is identical to the Nearest Neighbor Algorithm.
- However, it normalizes the attribute's ranges, which the NN algorithm can too.
- The IB1 Algorithm processes instances incrementally, unlike the traditional NN algorithm.

IB1 Algorithm…How the Concept Description Changes

- We will look at how IB1's concept description changes over time.
- This requires how we can use the similarity and classification functions of IB1 to yield an *extensional* concept description, although it is not necessary to produce the extensional CD.

IB1 Algorithm…How the Concept Description Changes…

- *Example*: A concept defined by two numeric dimensions where 100 training instances are randomly selected from a uniform distribution and the target concept consists of 4 disjuncts.
- The predicted boundaries of the target concept, drawn using solid lines, form a Voronoi diagram, and completely describes the classification prediction.
- Each predicted boundary of the target concept lies halfway between a pair of adjacent positive and negative instances.
- All instances on the same side of a boundary as a positive instance are predicted to be positive. All other instances are predicted to be negative.
- The shaded boundaries show the components of a concept to be learned. The dark boundaries that are learnt are approximations of the actual concept components.

IB1 Algorithm…How the Concept Description Changes…

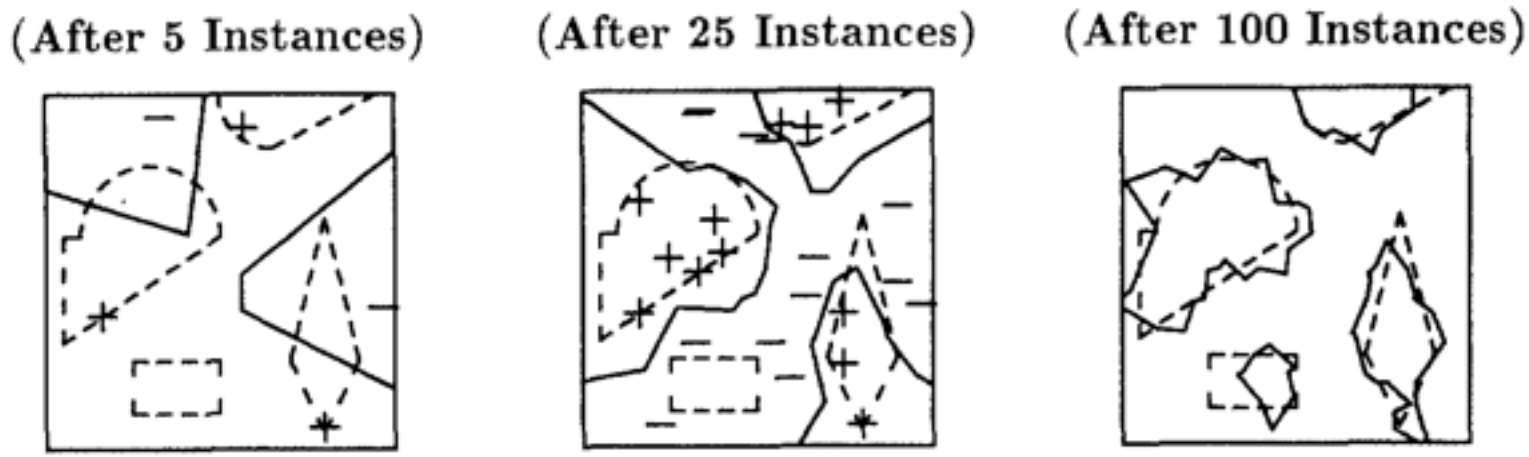


Figure 1. The extension of IBl's concept description, denoted by the solid lines, improves with training. Dashed lines delineate the four disjuncts of the target concept. Positive (+) and negative (−) instances are shown where possible.

IB1 Algorithm…How the Concept Description Changes…

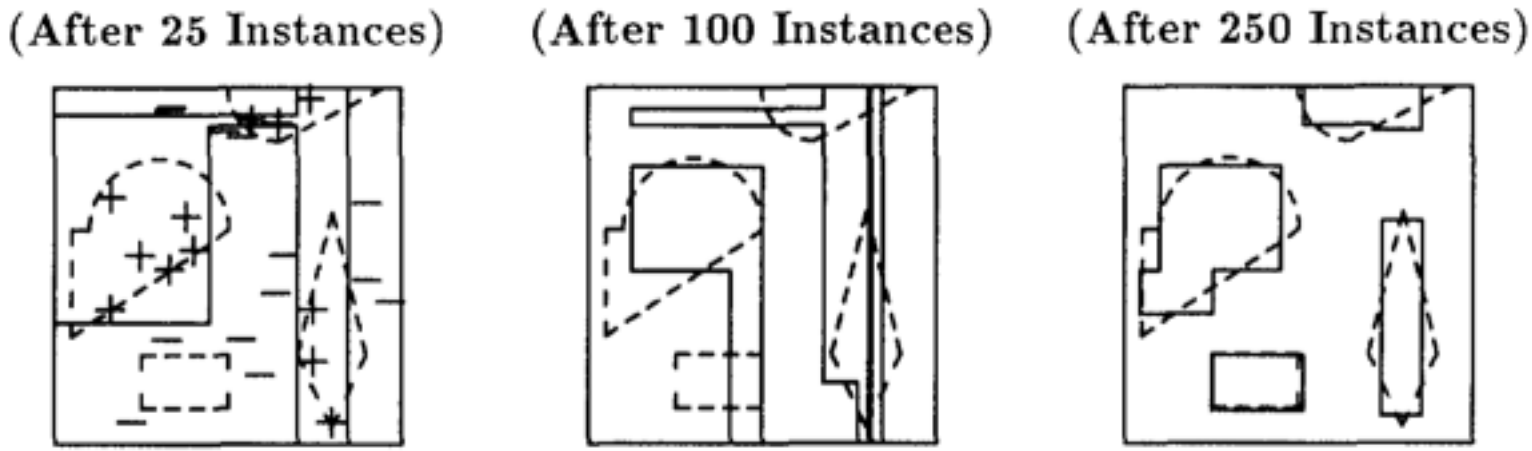


Figure 2. C4's extension eventually converges to an approximation similar to IB1's on this same training set, although C4's significance testing slows its learning rate.

IB1 Algorithm…

- We can see that the concept description changes over time.
- The concept description is disjunctive similar to what we saw in the case of decision trees. However, the concept description does not have to be always axis-aligned hyper-rectangles.
- IB1 does not make any assumption about the convexity of the target concept, its connectedness (the number of components or disjuncts), nor the relative positions of the various components of the target concept.
- Small perturbations in the shape of a concept are not captured.

Reducing Storage Requirements: IB2 Algorithm

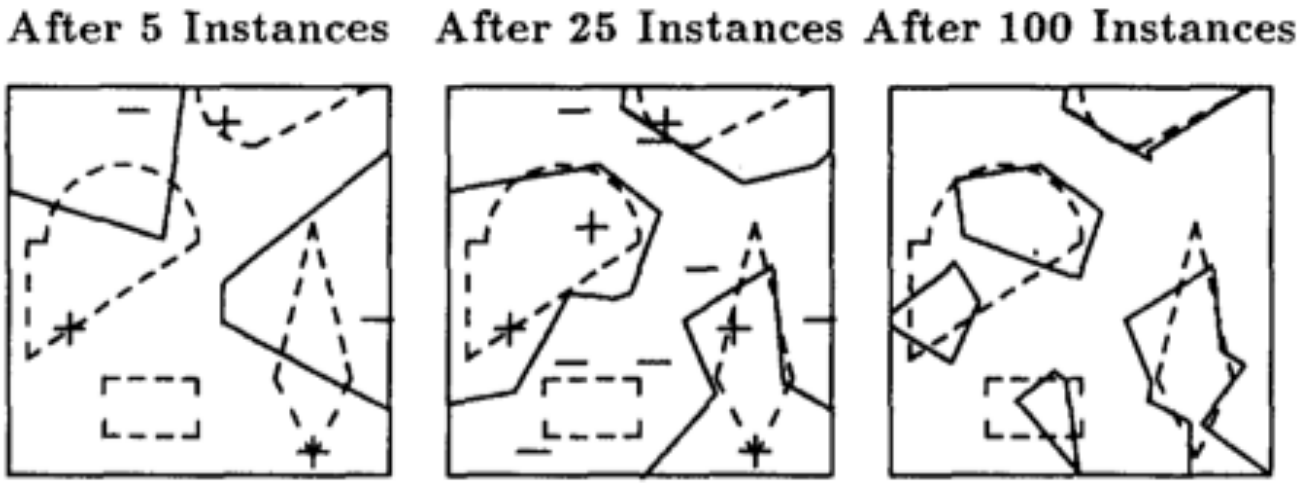
- It is not necessary to store every instance the learner has seen to classify unseen instances.
- Only instances near the boundaries in a small neighborhood of the boundary line, are needed to produce an accurate approximation of the concept boundary.
- The other instances do not distinguish where the concept boundary lies. In other words, instances away from the concept boundary do not really matter in classification.
- Therefore, we can save space by storing only informative instances.
- The set of informative instances can be approximated by by looking at the instances that IB1 misclassifies.

IB2 Algorithm

$CD \leftarrow \emptyset$
for each $x \in$ Training Set **do**
1. **for each** $y \in CD$ **do**
   Sim[y] $\leftarrow$ Similarity(x, y)
2. $y_{max} \leftarrow$ some $y \in CD$ with maximal Sim[y]
3. **if** class(x) = class(y_{max})
   then classification $\leftarrow$ **correct**
   else
   3.1 classification $\leftarrow$ **incorrect**
   3.2 $CD \leftarrow CD \cup \{x\}$

- IB2 is similar to IB1 except that it saves only misclassified instances.
- The intuition behind IB2 is that the vast majority of misclassified instances are near-boundary instances that are located in a small narrow neighborhood of the boundary, and these misclassified instances are outside the definition of the so-called core concept.

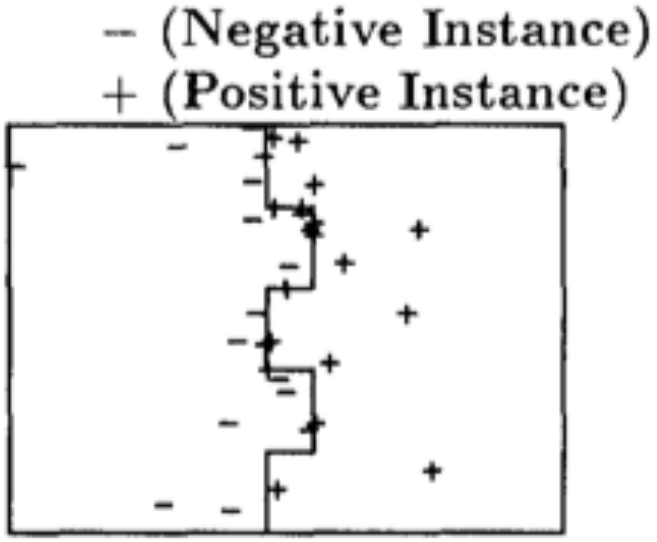
IB2 Algorithm's CD Improves Over Time Also



IB2's concept description also improves with training.

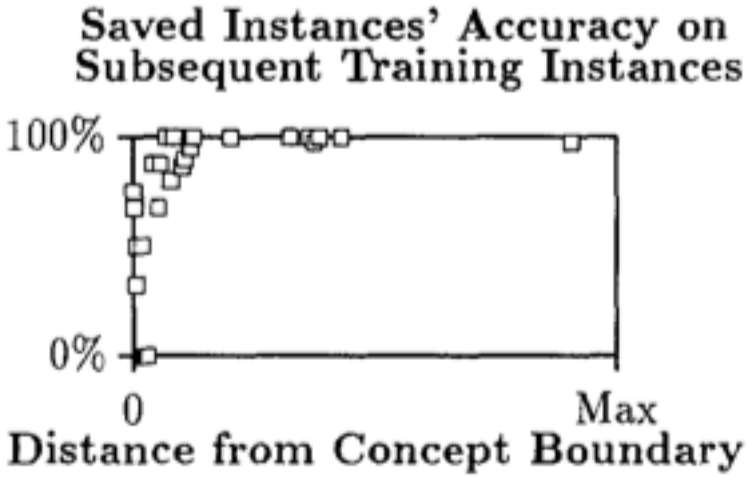
The paper concludes that IB2's approximation after 100 instances is almost as good as IB1's approximation after the same number of instances.

IB2 Saves Mostly Instances Near the Concept Boundary



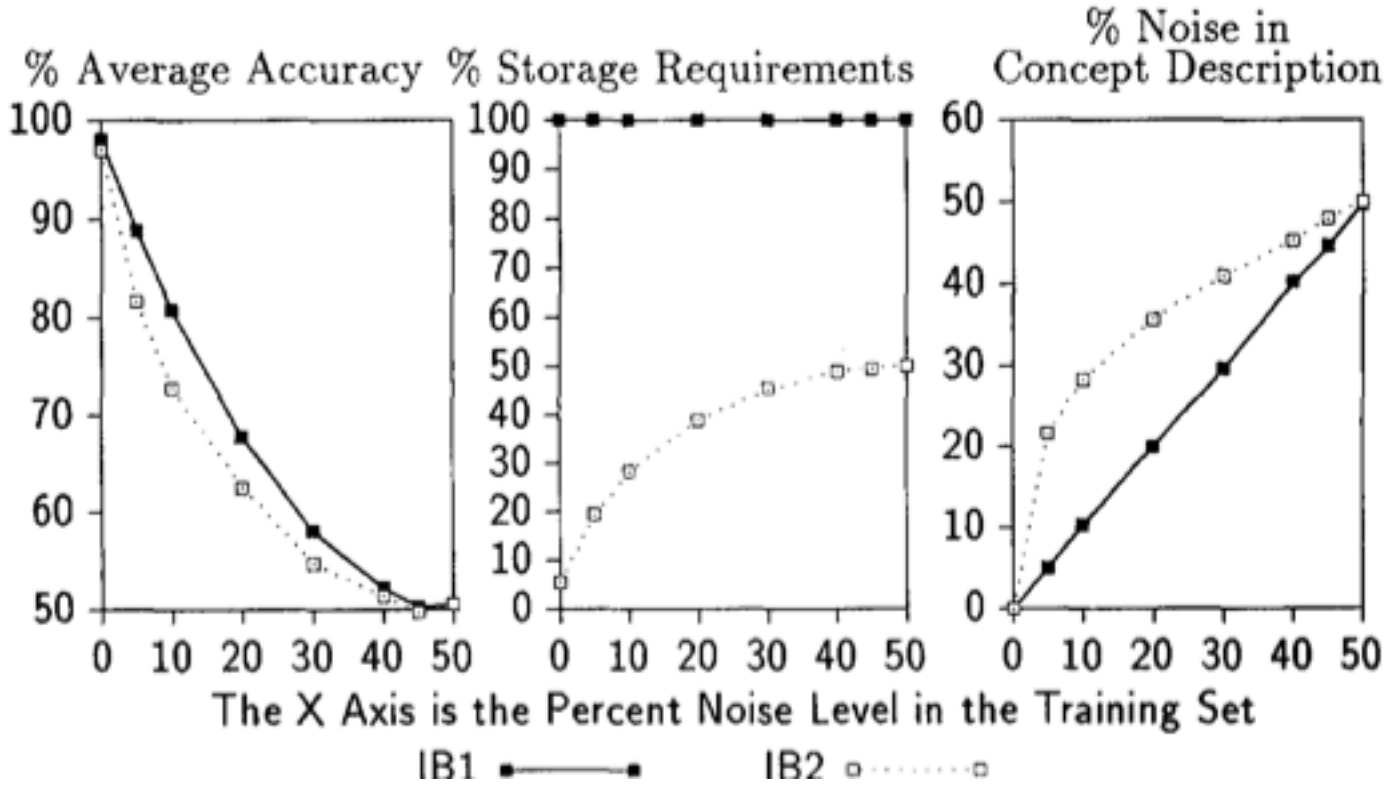
The saved instances are near the jagged concept boundary. This is a simpler classification problem, used to illustrate the instances saved by IB2.

IB2 Classifies Better Away from Concept Boundary



We see that the accuracy of subsequent training instances increases exponentially with distance from the concept boundary.

IB2's Behavior on a Simple Domain

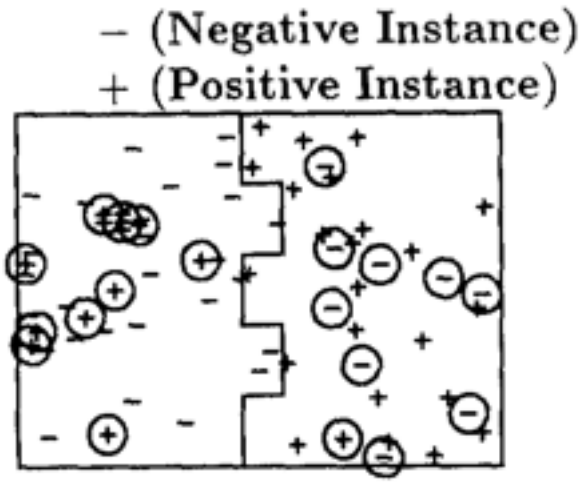


- IB2's storage requirement can be significantly lower than IB1, especially when the instances vary greatly in their distance from the concept boundary.
- The storage requirement is sensitive to noise.

IB2's Behavior in a Simple Domain

- Noise Level: It is the probability that each attribute value (except the class attribute) of each instance is replaced by a randomly-selected value from the attribute's domain, according to the attribute's distribution.
- IB2's classification accuracy decreases more quickly than does IB1's as the level of noise increases.
- This is because noisy instances are more likely to be misclassified and IB2 saves only misclassified instances, which it uses to generate classification decisions, in turn.

Which Instances Does IB2 Save?



- The figure shows saved instances by IB2 when the training set was corrupted with 10% noise level.
- In this case, 26.3% of IB2's saved instances were noisy.
- The circled ones are the noisy instances saved to describe a concept.

Empirical Studies with IB2

Table 3. Database characteristics.

Database Name	Training Set Size	Test Set Size	Number of Attributes	Number of Classes
Voting	350	85	16	2
Primary Tumor	275	64	17	22
LED Display	200	500	7	10
Waveform	300	500	21	3
Cleveland	250	53	13	2
Hungarian	250	44	13	2

Table 4. Percent accuracy ± standard error and percent storage requirements.

Database	IB1		IB2	
Voting	91.8 ± 0.4	100	90.9 ± 0.5	11.1
Tumor	34.7 ± 0.8	100	32.9 ± 0.8	71.3
LED Display	70.5 ± 0.4	100	62.4 ± 0.6	41.5
Waveform	75.2 ± 0.3	100	69.6 ± 0.4	32.5
Cleveland	75.7 ± 0.8	100	71.4 ± 0.8	30.4
Hungarian	58.7 ± 1.5	100	55.9 ± 2.0	36.0

Empirical Studies with IB2…

- IB2 can reduce IB1's storage requirements significantly.
- When the dataset is relatively noise-free, IB2's classification accuracies are comparable to IB1's.
- If the dataset is noisy, IB2's accuracy is a little lower.
- The datasets used:
  - *Noise-free Congressional Voting Dataset*: Noise-free, linearly separable, 16 boolean attributes, 288 missing values in 435 instances, categorial attribute: The party of a member. *IB2 performs equally well, with much lower storage requirement.*
  - *Sparse Primary Tumor Dataset*: Sparsely described (not many instances for concepts) concepts in high dimensional space. 11 concepts are described by less than 10 instances. *IB2 performs almost equally well, with a little lower storage. There are not enough instances in each concept to determine the concept boundaries well.*

Empirical Studies with IB2…

- The datasets used:
  - *Noisy artificial domains, LED Display and Waveform Datasets*: IB2's classification substantially worse than IB1's in such cases.
  - *Domains with imperfect attributes, Cleveland and Hungarian Datasets*: These are datasets where the concepts are not well-defined in terms of the attributes, as in medical diagnosis. IB2 takes significantly less space, but classification accuracy is substantially lower as well.

IB3: Noise-tolerant Extension to IB2

- IB2 is sensitive to the amount of noise present in the training set, but it reduces IB1's storage requirement drastically in many application domains.
- For example, it requires only 11.1% space in the Congressional Voting Dataset to make almost equal prediction.
- Maybe, a noise-tolerant version of IB2, that decides which of the saved instances should be used to make classification decision, using a simple test or filter, will make things better for IB2.

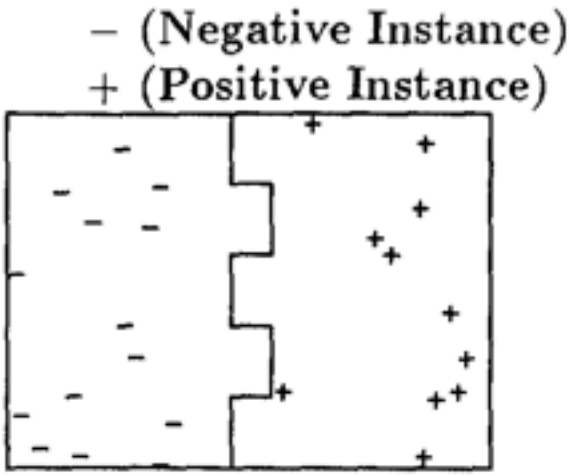
IB3 Algorithm

$CD \leftarrow \emptyset$
for each x in Training Set **do**
1. **for each** $y \in CD$ **do**
   Sim[y] $\leftarrow$ Similarity(x, y)
2. **if** $\exists\{y \in CD |$ acceptable(y)$\}$
   then $y_{max} \leftarrow$ some acceptable $y \in Cd$ with maximal Sim[y]
   else
      2.1 $i \leftarrow$ a randomly-selected value in [1,$|CD|$]
      2.2 $y_{max} \leftarrow$ some $y \in CD$ that is the i-th most similar instance to x
3. **if** class(x) $\neq$ class(y_{max})
   then classification $\leftarrow$ **correct**
   else
      3.1 classification $\leftarrow$ **incorrect**
      3.2 $CD \leftarrow CD \cup \{x\}$
4. **for each** y in CD **do**
   if Sim[y] $\geq$ Sim[y_{max}]
   then
      4.1 Update y's classification record
      4.2 **if** y's record is significantly poor
         then $CD \leftarrow C - \{y\}$

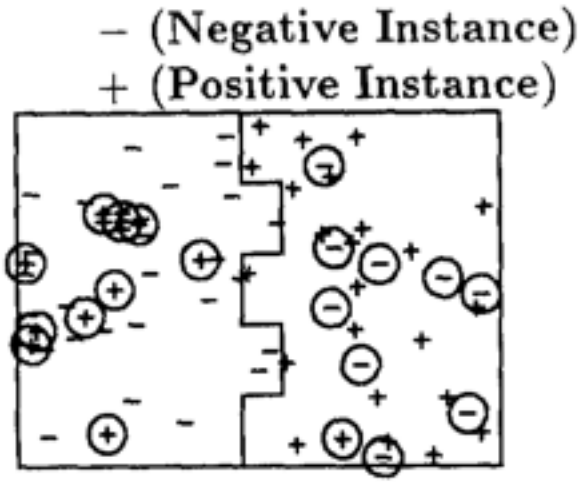
IB3 Algorithm…

- IB3 employs a "*wait and see*" evidence gathering method to determine which saved instances are expected to perform well during classification.
- IB3 maintains a *classification record* with each saved instance. A classification record saves an instance's classification performance on subsequently presented training instances and suggests how it will perform in the future.
- IB3 uses a *significance test* to determine which instances are good classifiers and which ones are believed to be noisy. The former are used to classify subsequently presented instances. The latter are discarded from the concept description.

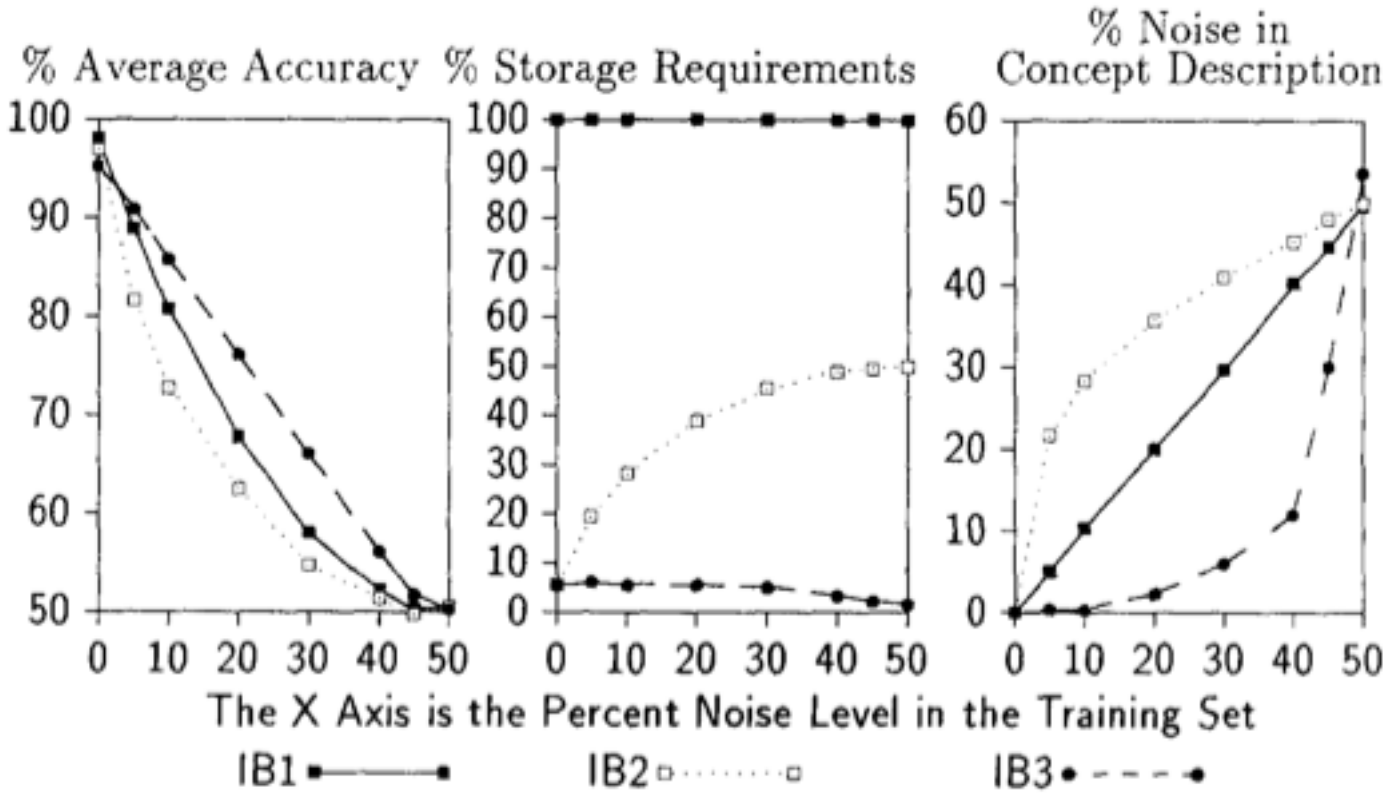
IB3 with Noise



IB3's saved instances with the noisy dataset that produced the saved instances shown below for IB2.



IB3 with Artificial Dataset



IB3 filters noise much better than IB2 or IB1. IB3 takes less space IB2.

Empirical Results with IB3

Table 6. Percent accuracy ± standard error and percent storage requirements. IB3 recorded higher classification accuracies and lower storage requirements than IB2. It also compares favorably with the other algorithms' classification accuracies.

Database	IB1	IB2	IB3	C4
Voting	91.8 ± 0.4 100	90.9 ± 0.5 11.1	91.6 ± 0.5 7.4	95.5 ± 0.3
Tumor	34.7 ± 0.8 100	32.9 ± 0.8 71.3	38.6 ± 0.9 16.4	37.8 ± 0.9
LED Display	70.5 ± 0.4 100	62.4 ± 0.6 41.5	71.7 ± 0.4 28.7	68.3 ± 0.3
Waveform	75.2 ± 0.3 100	69.6 ± 0.4 32.5	73.8 ± 0.4 14.6	70.7 ± 0.3
Cleveland	75.7 ± 0.8 100	71.4 ± 0.8 30.4	78.0 ± 0.8 7.7	75.5 ± 0.7
Hungarian	58.7 ± 1.5 100	55.9 ± 2.0 36.0	80.5 ± 0.9 7.5	78.2 ± 0.9

C4 is a decision-tree based classifier.

Limitations of IB3

- IB3 assumes concepts are disjoint and that they jointly exhaust the instance space.
- IB3 cannot represent overlapping concepts and it assume that every instance is a member of exactly one concept. This is can be overcome by learning a separate concept description for each concept.
- IB3's learning performance is highly sensitive to the number of irrelevant attributes used to describe instances.
- As the number of attributes (dimensionality) goes up, IB3's performance goes down drastically and its space requirement goes up also drastically. So, IB3 doesn't work well with high dimensional datasets with many irrelevant attributes.

Advantages of IB3

- Instance-based algorithms are *simple* to understand so that one can make a rigorous analysis of what works and what doesn't work.
- The inductive bias for IBL algorithms is not very restrictive or is relaxed. They can *incrementally* learn *piecewise linear approximations* of a concept. In contrast, decision trees learn hyper-rectangular representations with all the data.
- IBL algorithms can update concept descriptions efficiently.
- IBL leads to robust learning algorithms. They can tolerate noise and irrelevant attributes and can represent probabilistic and overlapping concepts (one needs some extensions for each of these).