Some of ML Advances

JERZY STEFANOWSKI
Institute of Computing Sciences
Poznań University of Technology

Wykład podsumowujący: 2010
Oparte na różnych moich wykładach

List of „absence” in this year course

Advances in supervised learning

1. Multiple classifiers (ensembles)
2. Imbalanced learning
3. Passive vs. active learning
4. Incremental on-line learning (and concept drift)

Semi-supervised learning

Multistrategic learning

Knowledge more instensive learning

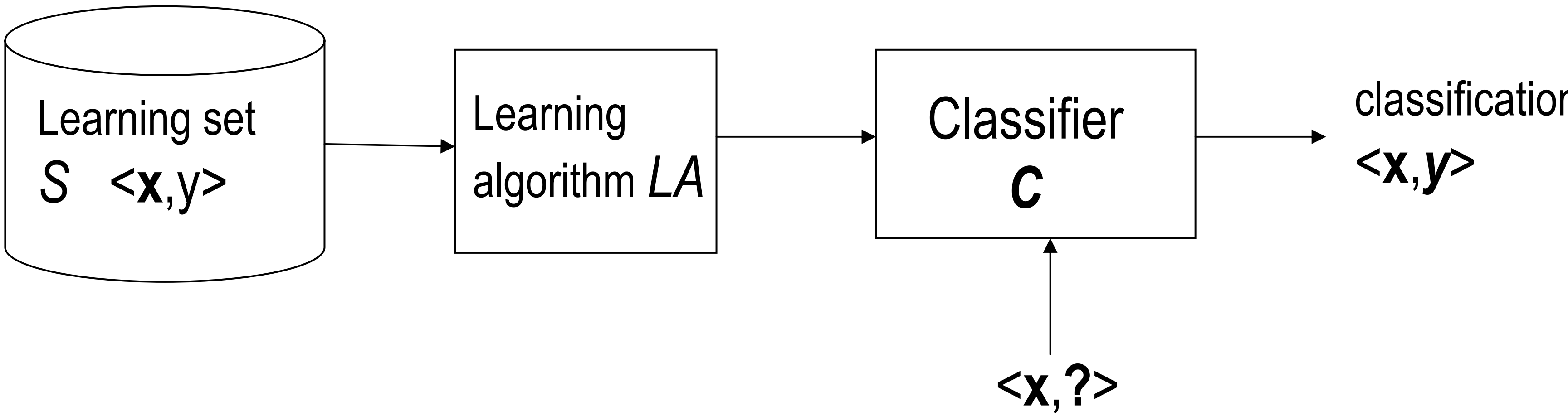
1. Inductive logic learning
2. n^2 classifier for multi-class problems

Reinforcement Learning

Theory of Learning (COLT, PAC, VC-dimensions)

Typical Schema for Supervised Learning of Classification

Supervised Learning of Classification - assigning a decision class label to a set of objects described by a set of attributes



Set of learning examples S = $\{\langle \mathbf{x}_1, y_1\rangle, \langle \mathbf{x}_2, y_2\rangle, \cdots; \langle \mathbf{x}_n, y_n\rangle\}$

for some unknown classification function f: $y = f(\mathbf{x})$

$\mathbf{x}_i = <x_{i1}, x_{i2}, \ldots, x_{im}>$ example described by m attributes

y – class label; value drawn from a discrete set of classes $\{Y_1, \ldots, Y_K\}$

Why could we integrate classifiers?

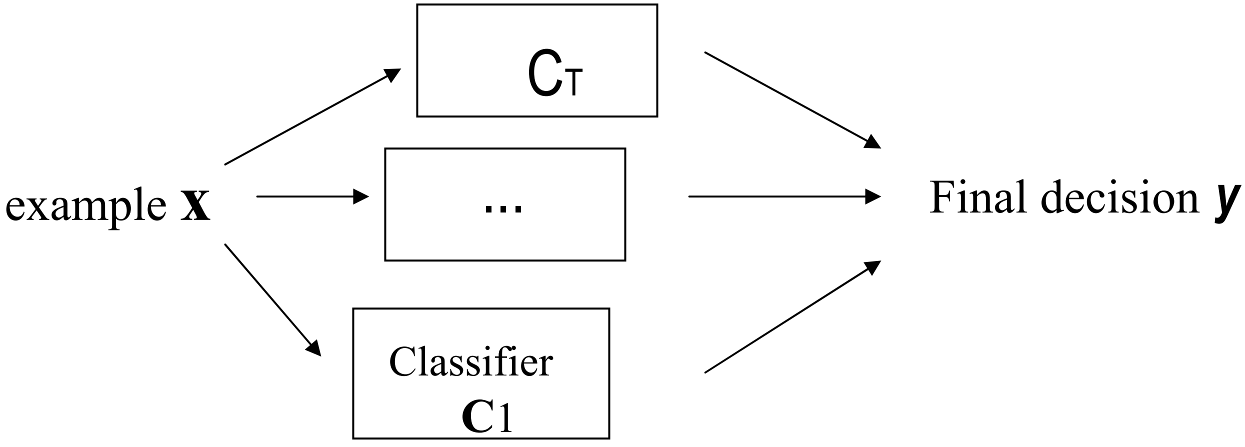
- Typical research $\rightarrow$ create and evaluate a single learning algorithm; compare performance of some algorithms.
- Empirical observations or applications $\rightarrow$ a given algorithm may outperform all others for a specific subset of problems
  - There is no one algorithm achieving the best accuracy for all situations! [No free lunch]
- A complex problem can be decomposed into multiple sub-problems that are easier to be solved.
- Growing research interest in combining a set of learning algorithms / classifiers into one system

„Multiple learning systems try to exploit the local different behavior of the base learners to enhance the accuracy of the overall learning system”

- G. Valentini, F. Masulli

Multiple classifiers / ensembles - definitions

- Multiple classifier – a set of classifiers whose individual predictions are combined in some way to classify new examples.
- Various names: ensemble methods, committee, classifier fusion, combination, aggregation,…
- Integration should improve predictive accuracy.



Multiple classifiers – why do they work?

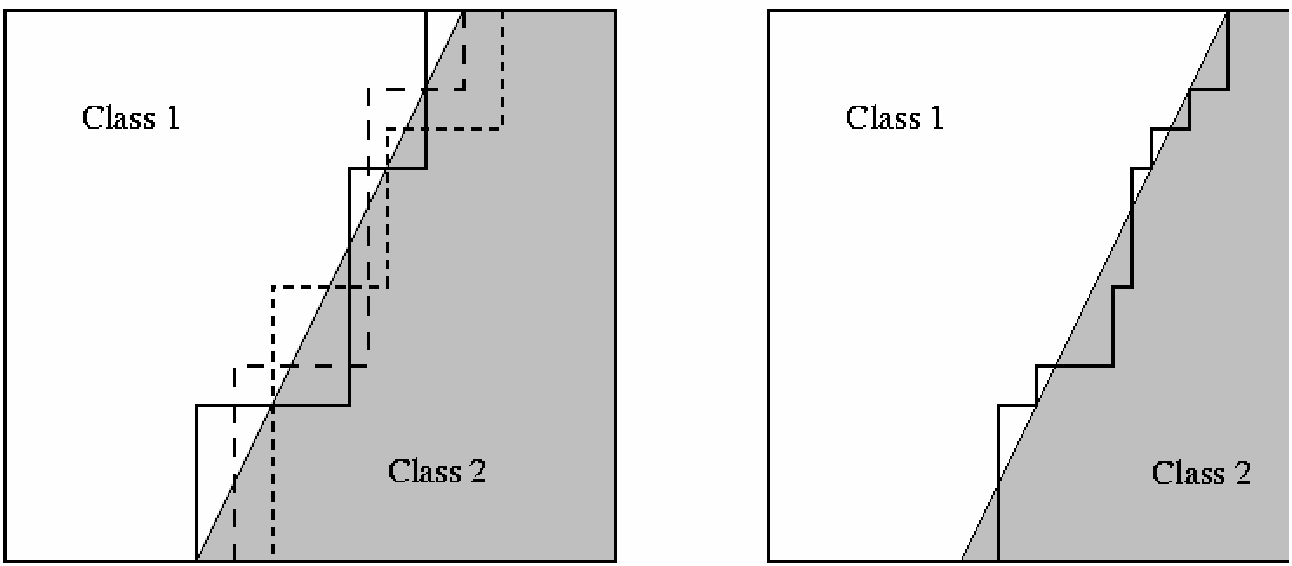
- How to create such systems and when they may perform better than their components used independently?
- Combining identical classifiers is useless!

> A necessary condition for the approach to be useful is that member classifiers should have a substantial level of disagreement, i.e., they make error independently with respect to one another

- Conclusions from some studies (e.g. Hansen&Salamon90, Ali&Pazzani96):
  Member classifiers should **make uncorrelated errors** with respect to one another; each classifier should perform better than a random guess.

Multiple classifier may work better than a single classifier.

- The diagonal decision boundary may be difficult for individual classifiers, but may be approximated by ensemble averaging.
- Decision boundaries constricted by decision trees → hyperplanes parallel to the coordinate axis - „staircases”.
- By averaging a large number of „staircases” the diagonal boundary can be approximated with some accuracy.



Combing classifier predictions

- Intuitions:
  - Utility of combining diverse, independent opinions in human decision-making
- Voting vs. non-voting methods
  - Counts of each classifier are used to classify a new object
  - The vote of each classifier may be weighted, e.g., by measure of its performance on the training data. (Bayesian learning interpretation).
- Non-voting $\rightarrow$ output classifiers (class-probabilities or fuzzy supports instead of single class decision)
  - Class probabilities of all models are aggregated by specific rule (product, sum, min, max, median,…)
  - More complicated $\rightarrow$ extra meta-learner

Group or specialized decision making

- Group (static) – all base classifiers are consulted to classify a new object.
- Specialized / dynamic integration – some base classifiers performs poorly in some regions of the instance space
  - So, select only these classifiers whose are „expertised" (more accurate) for the new object

Diversification of classifiers

- <u>Different training sets</u> (different samples or splitting,..)
- <u>Different classifiers</u> (trained for the same data)
- Different attributes sets

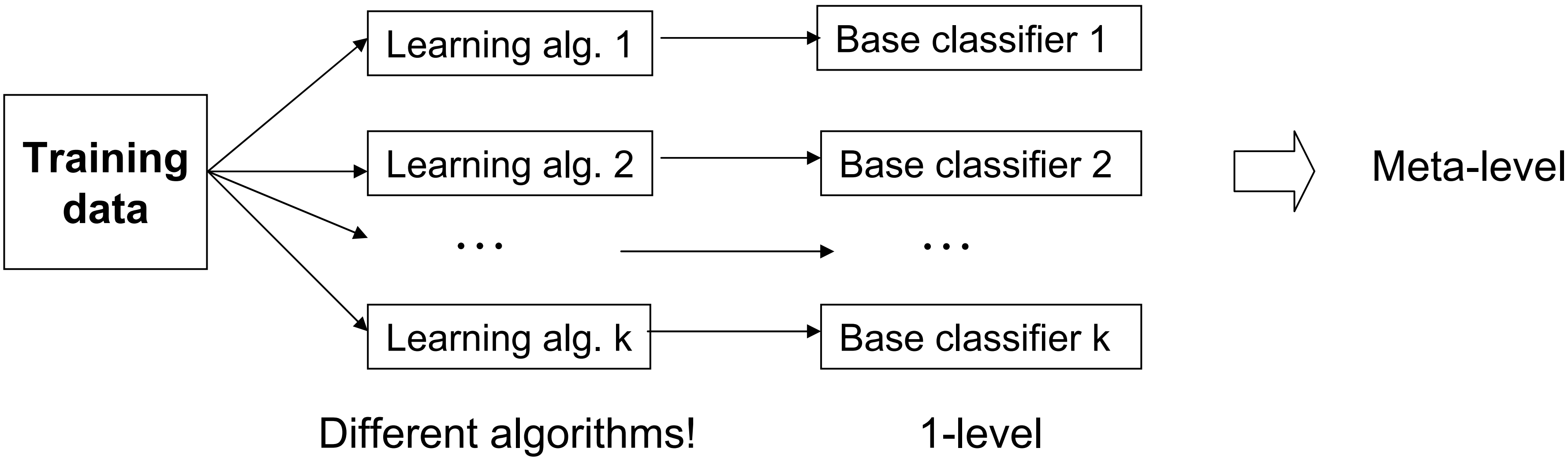
  (e.g., identification of speech or images)
- Different parameter choices

  (e.g., amount of tree pruning, BP parameters, number of neighbors in KNN,…)
- Different architectures (like topology of ANN)
- Different initializations

Stacked generalization [Wolpert 1992]

- Use meta learner instead of averaging to combine predictions of base classifiers.
  - Predictions of base learners (*level-0 models*) are used as input for meta learner (*level-1 model)*
- Method for generating base classifiers usually apply different learning schemes.
- Hard to analyze theoretically.

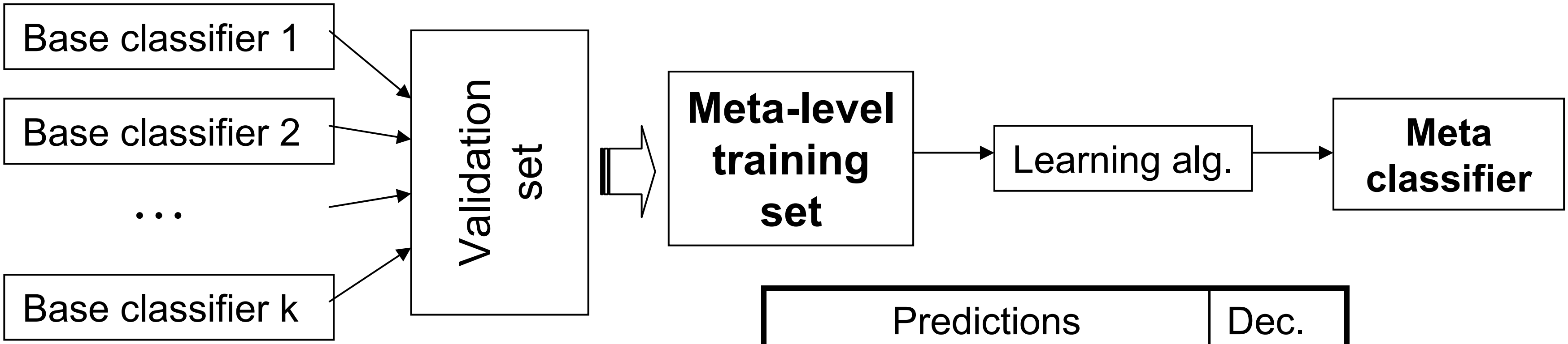
The Combiner - 1



Chan & Stolfo : *Meta-learning*.

- Two-layered architecture:
  - 1-level – base classifiers.
  - 2-level – meta-classifier.
- Base classifiers created by applying the different learning algorithms to the same data.

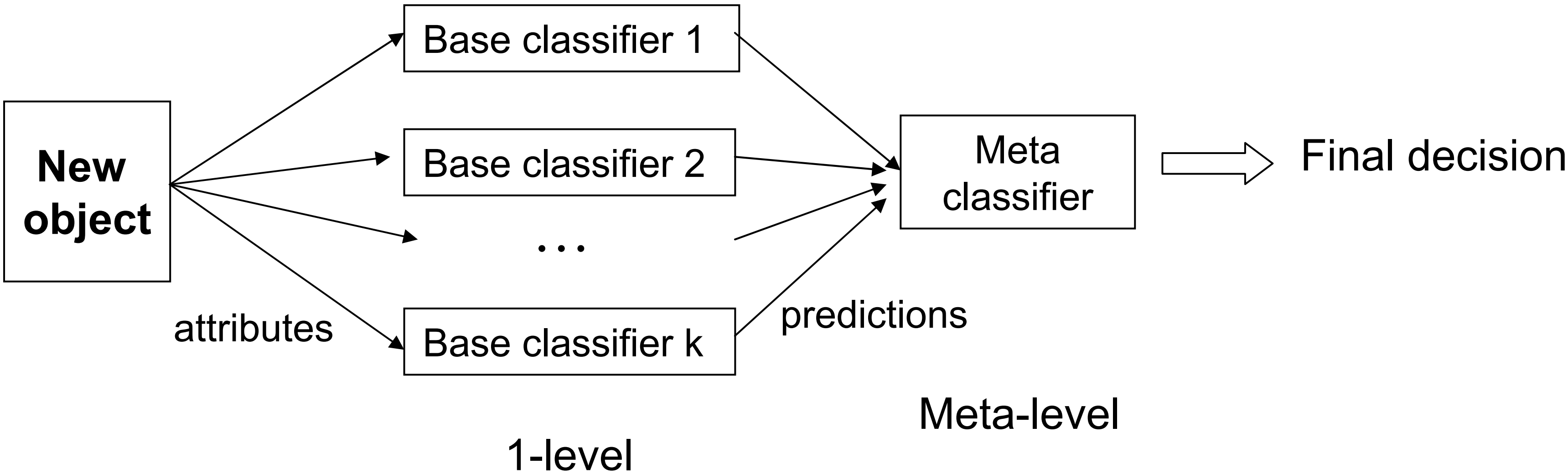
Learning the meta-classifier



Predictions				Dec. class
Cl.1	Cl.2	…	Cl.K	
A	A	…	B	**A**
A	B	…	C	**B**

- Predictions of base classifiers on an extra validation set (not directly training set – apply „internal” cross validation) with correct class decisions → a meta-level training set.
- An extra learning algorithm is used to construct a meta-classifiers.
- The idea → a meta-classifier attempts to learn relationships between predictions and the final decision;
  It may correct some mistakes of the base classifiers.

The Combiner - 2



Classification of a new instance by the combiner

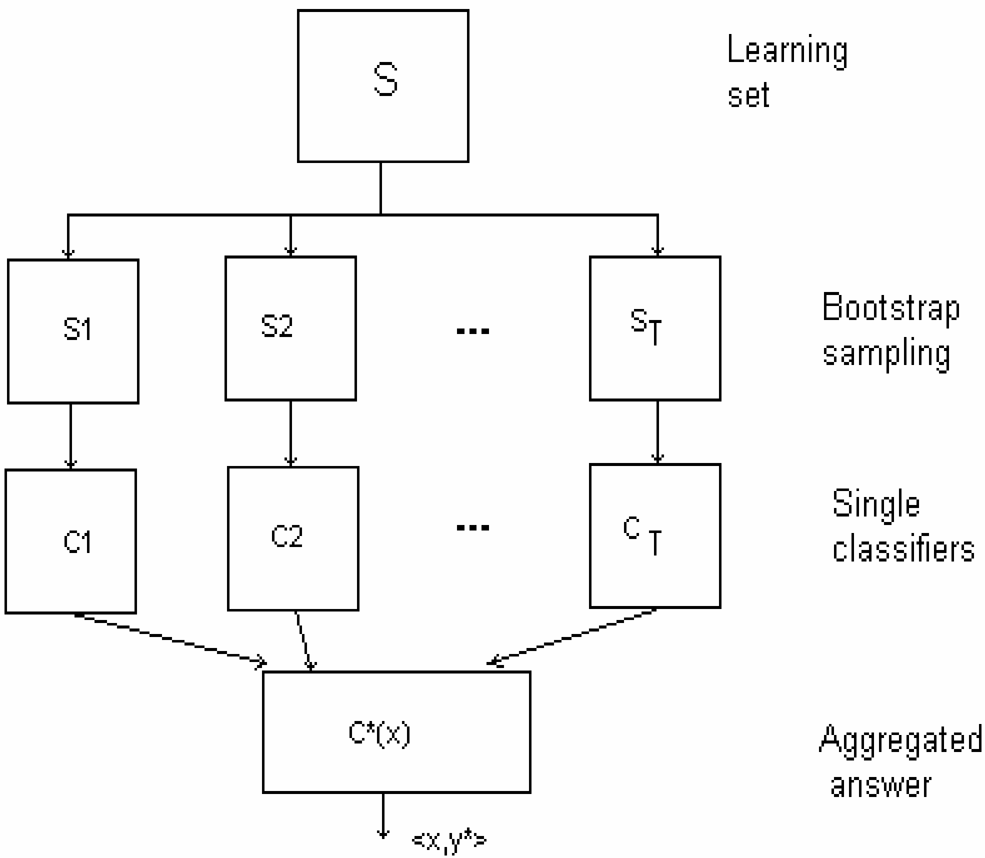
- Chan & Stolfo [95/97] : experiments that their combiner ({CART,ID3,K-NN}$\rightarrow$NBayes) is better than equal voting.

Bagging [L.Breiman, 1996]

- Bagging = **B**ootstrap **agg**regation
  - Generates individual classifiers on bootstrap samples of the training set
- As a result of the sampling-with-replacement procedure, each classifier is trained on the average of 63.2% of the training examples.
  - For a dataset with N examples, each example has a probability of $1-(1-1/N)^N$ of being selected at least once in the N samples. For $N \rightarrow \infty$, this number converges to (1-1/e) or 0.632 [Bauer and Kohavi, 1999]
- Bagging traditionally uses component classifiers of the same type (e.g., decision trees), and combines prediction by a simple majority voting across.

More about „Bagging”

- Bootstrap aggregating – L.Breiman [1996]



input S – learning set, T – no. of bootstrap samples, LA – learning algorithm

output C* - multiple classifier

for i=1 **to** T **do**

begin

  S_i:=bootstrap sample from S;

  C_i:=$LA(S_i)$;

end;

$$C^*(x) = \operatorname{argmax}_y \sum_{i=1}^{T} (C_i(x) = y)$$

Bagging Empirical Results

Misclassification error rates [Percent]

Data	Single	Bagging	Decrease
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	22%
soybean	14.5	10.6	27%

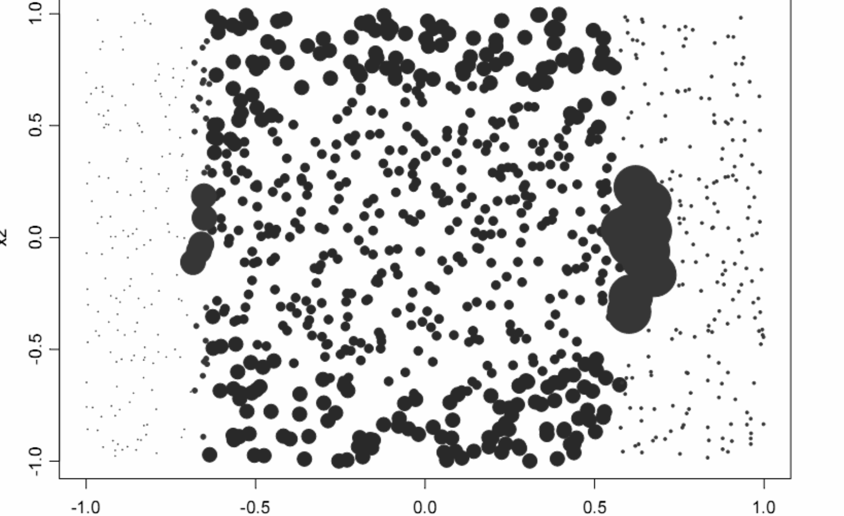
Breiman "Bagging Predictors" Berkeley Statistics Department TR#421, 1994

Boosting [Schapire 1990; Freund & Schapire 1996]

- In general takes a different weighting schema of resampling than bagging.
- Freund & Schapire: theory for "weak learners" in late 80's
- **Weak Learner**: performance on ***any*** train set is slightly better than chance prediction
  - Schapire has shown that a weak learner can be converted into a strong learner by changing the distribution of training examples
- Iterative procedure:
  - The component classifiers are built sequentially, and examples that are misclassified by previous components are chosen more often than those that are correctly classified!
  - So, new classifiers are influenced by performance of previously built ones. New classifier is encouraged to become expert for instances classified incorrectly by earlier classifier.
- There are several variants of this algorithm – **AdaBoost** the most popular (see also arcing).

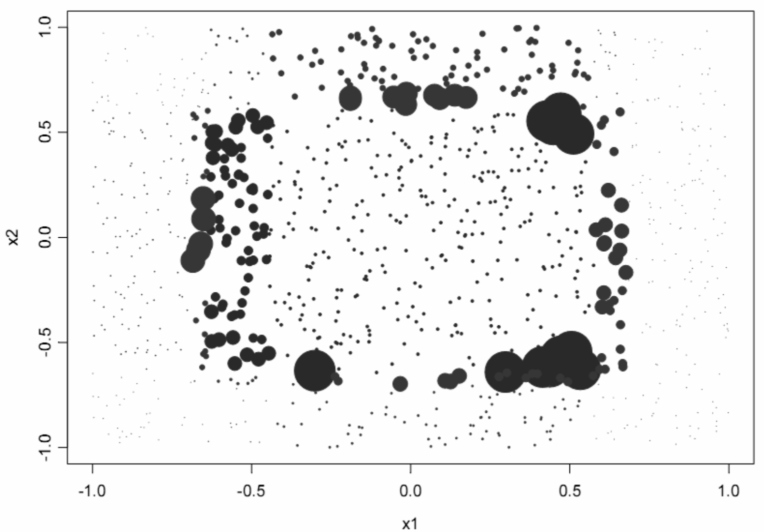
AdaBoost

- Weight all training examples equally (1/n)
- Train model (classifier) on train sample D_i
- Compute error e_i of model on train sample D_i
- A new training sample D_{i+1} is produced by decreasing the weight of those examples that were correctly classified (multiple by $e_i/(1-e_i)$)), and increasing the weight of the misclassified examples.
- Normalize weights of all instances.
- Train new model on re-weighted train set
- Re-compute errors on weighted train set
- The process is repeated until (# iterations or error stopping)
- Final model: weighted prediction of each classifier
  - Weight of class predicted by component classifier $\log(e_i/(1-e_i))$

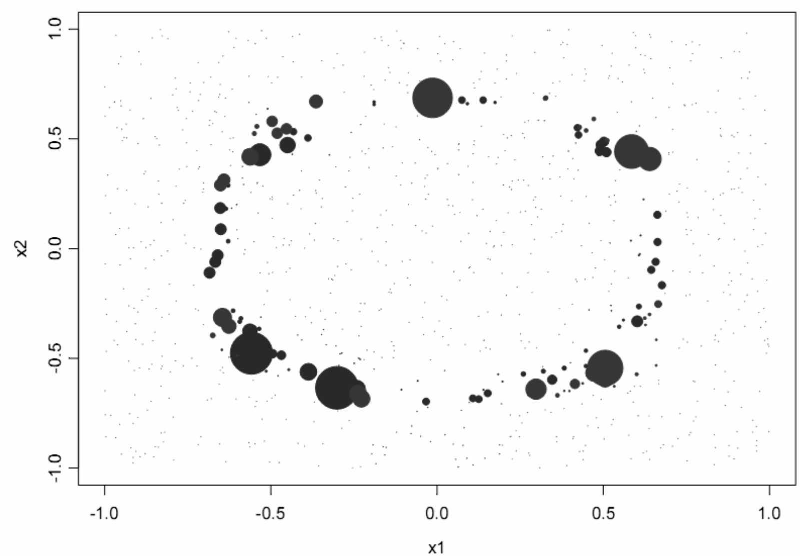


Classifications (colors) and
Weights (size) after *1 iteration*
Of AdaBoost

iterations



20 iterations



from Elder, John. From Trees to Forests and Rule Sets - A Unified Overview of Ensemble Methods. 2007.

Boosting vs. Bagging with C4.5 [Quinlan 96]

	C4.5	Bagged C4.5 vs C4.5			Boosted C4.5 vs C4.5			Boosting vs Bagging	
	err (%)	err (%)	w-l	ratio	err (%)	w-l	ratio	w-l	ratio
anneal	7.67	6.25	10-0	.814	4.73	10-0	.617	10-0	.758
audiology	22.12	19.29	9-0	.872	15.71	10-0	.710	10-0	.814
auto	17.66	19.66	2-8	1.113	15.22	9-1	.862	9-1	.774
breast-w	5.28	4.23	9-0	.802	4.09	9-0	.775	7-2	.966
chess	8.55	8.33	6-2	.975	4.59	10-0	.537	10-0	.551
colic	14.92	15.19	0-6	1.018	18.83	0-10	1.262	0-10	1.240
credit-a	14.70	14.13	8-2	.962	15.64	1-9	1.064	0-10	1.107
credit-g	28.44	25.81	10-0	.908	29.14	2-8	1.025	0-10	1.129
diabetes	25.39	23.63	9-1	.931	28.18	0-10	1.110	0-10	1.192
glass	32.48	27.01	10-0	.832	23.55	10-0	.725	9-1	.872
heart-c	22.94	21.52	7-2	.938	21.39	8-0	.932	5-4	.994
heart-h	21.53	20.31	8-1	.943	21.05	5-4	.978	3-6	1.037
hepatitis	20.39	18.52	9-0	.908	17.68	10-0	.867	6-1	.955
hypo	.48	.45	7-2	.928	.36	9-1	.746	9-1	.804
iris	4.80	5.13	2-6	1.069	6.53	0-10	1.361	0-8	1.273
labor	19.12	14.39	10-0	.752	13.86	9-1	.725	5-3	.963
letter	11.99	7.51	10-0	.626	4.66	10-0	.389	10-0	.621
lymphography	21.69	20.41	8-2	.941	17.43	10-0	.804	10-0	.854
phoneme	19.44	18.73	10-0	.964	16.36	10-0	.842	10-0	.873
segment	3.21	2.74	9-1	.853	1.87	10-0	.583	10-0	.684
sick	1.34	1.22	7-1	.907	1.05	10-0	.781	9-1	.861
sonar	25.62	23.80	7-1	.929	19.62	10-0	.766	10-0	.824
soybean	7.73	7.58	6-3	.981	7.16	8-2	.926	8-1	.944
splice	5.91	5.58	9-1	.943	5.43	9-0	.919	6-4	.974
vehicle	27.09	25.54	10-0	.943	22.72	10-0	.839	10-0	.889
vote	5.06	4.37	9-0	.864	5.29	3-6	1.046	1-9	1.211
waveform	27.33	19.77	10-0	.723	18.53	10-0	.678	8-2	.938
average	*15.66*	*14.11*		*.905*	*13.36*		*.847*		*.930*

Table 1: Comparison of C4.5 and its bagged and boosted versions.

Boosting vs. Bagging

- Bagging doesn't work so well with stable models. Boosting might still help.
- Boosting might hurt performance on noisy datasets. Bagging doesn't have this problem.
- On average, boosting helps more than bagging, but it is also more common for boosting to hurt performance.
- In practice bagging almost always helps.
- Bagging is easier to parallelize.

Feature-Selection Ensembles

- ***Key idea:*** Provide a different subset of the input features in each sample and call of the learning algorithm.
  - ***Example:*** Venus&Cherkauer (1996) trained an ensemble with 32 neural networks. The 32 networks were based on 8 different subsets of 119 available features and 4 different algorithms. The ensemble was significantly better than any of the neural networks!
- See also Random Subspace Methods by Ho.
- Integrating attribute selection with bagging
  - Increasing diversification of component classifiers
  - Boostrap sample like in bagging + random selection of attributes
    - Study of P.Latinne *et al*. → encouraging results of simple random technique (BagFS, Bag vs. MFS)
  - My and M.Kaczmarek study → we have used different techniques of attribute subset selection – also improves accuracy

Random forests [Breiman 2001]

- At every level of the tree, choose a random subset of the attributes (not examples) and choose the best split among those attributes.
- Combined with selecting examples like basic bagging.
- Doesn't overfit.

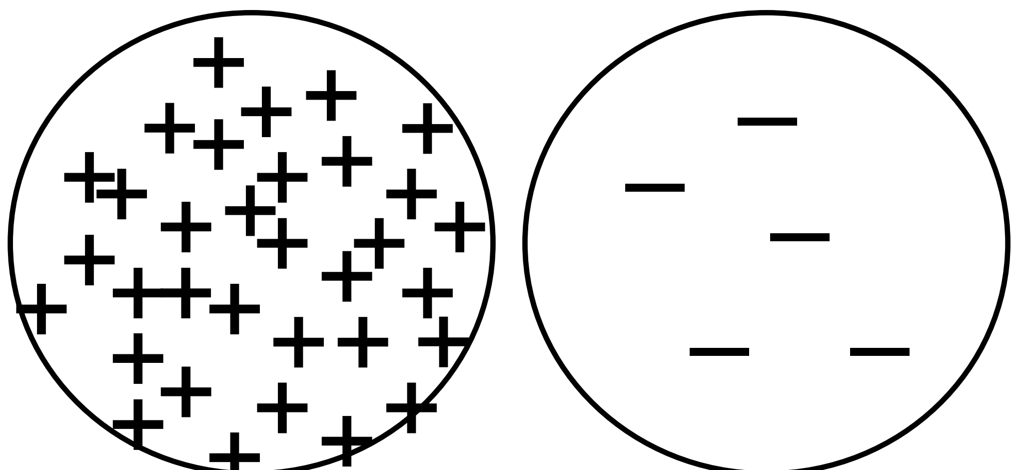
Data set	Adaboost	Selection	Forest-RI single input	One
Glass	22.0	20.6	21.2	36
Breast cancer	3.2	2.9	2.7	6
Diabetes	26.6	24.2	24.3	33
Sonar	15.6	15.9	18.0	31
Vowel	4.1	3.4	3.3	30
Ionosphere	6.4	7.1	7.5	12
Vehicle	23.2	25.8	26.4	33
German credit	23.5	24.4	26.2	33
Image	1.6	2.1	2.7	6
Ecoli	14.8	12.8	13.0	24
Votes	4.8	4.1	4.6	7
Liver	30.7	25.1	24.7	40
Letters	3.4	3.5	4.7	19
Sat-images	8.8	8.6	10.5	17
Zip-code	6.2	6.3	7.8	20
Waveform	17.8	17.2	17.3	34
Twonorm	4.9	3.9	3.9	24
Threenorm	18.8	17.5	17.5	38
Ringnorm	6.9	4.9	4.9	25

eiman, Leo (2001). "Random Forests". Machine Learning 45 (1), 5-32

Learning from Imbalanced Data

Introductory remarks

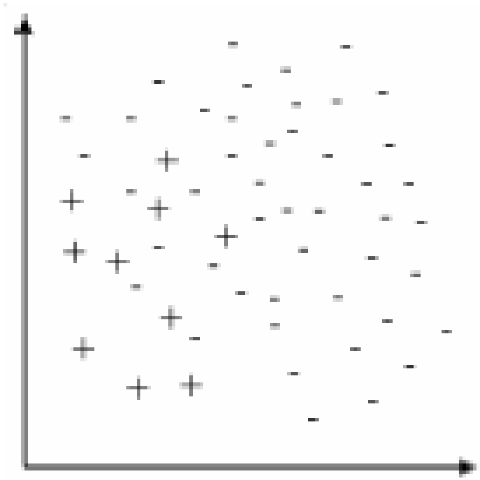
- A data set is imbalanced if the classes are not approximately equally represented.
  - One class (a minority class) includes much smaller number of examples than other classes.
- Rare examples /class are often of special interest
- Typical problems in medical or technical diagnostics, finance, image recognition, telecommunication networks, document filtering.
- CLASS IMBALANCE → causes difficulties for learning and decrease the classifier performance.



Class imbalance is not the same
s COST sensitive learning.
n general costs are unknown!

Imbalance → Difficulties for Learning Classifiers

- Standard approach to learn classifiers are designed under assumption of partly balanced classes and to optimize overall accuracy without taking into account the relative distribution of each class.
  - As a result, these classifiers tend to ignore small classes while concentrating on classifying the large ones accurately
- Imbalance ratio is not the main problem
- Some other sources of difficulties:
  - Rare data and small disjuncts
  - Ambiguous boundary between classes
  - Influence of noisy examples



Taxonomy of approach

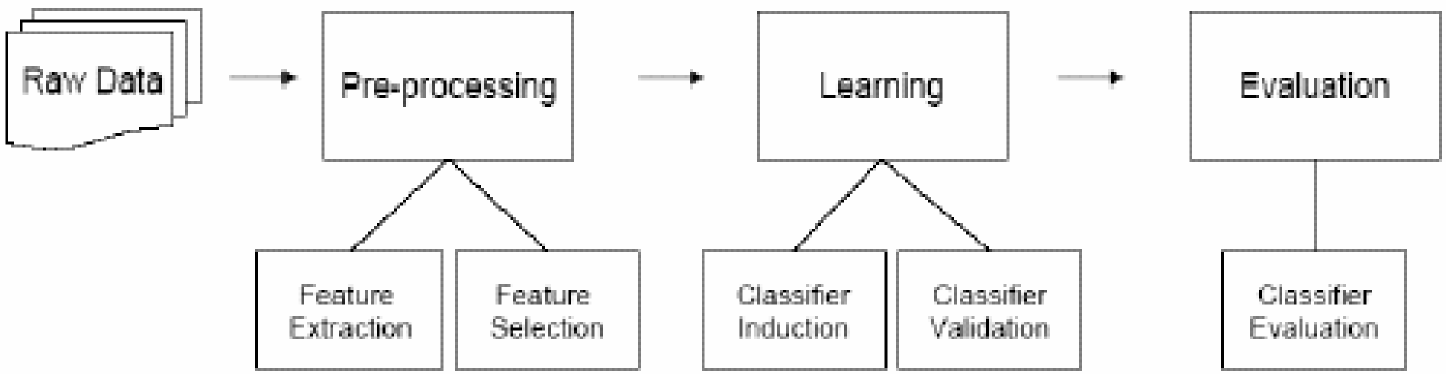
- Review survey, e.g.,
  - Weiss G.M., Mining with rarity: a unifying framework. ACM Newsletter, 2004.
- Main approaches to deal with imbalance of data:
  - Data or algorithmic level
    - Re-sampling or re-weighting,
    - Changing search strategies in learning,
      use another measures,
    - Adjusting classification strategies,
    - One-class-learning
    - Using hybrid and combined approaches (boosting like re-weighing)
    - …

Passive vs. Active Learning

+ Semi-supervised paradigms

A typical approach to supervised learning

- Construct data representation (objects x attributes) and label examples
- Possibly pre-process (feature construction)
- Learn from all labeled examples
- Access to all training data!



Motivations for Learning from Partially Labeled Examples

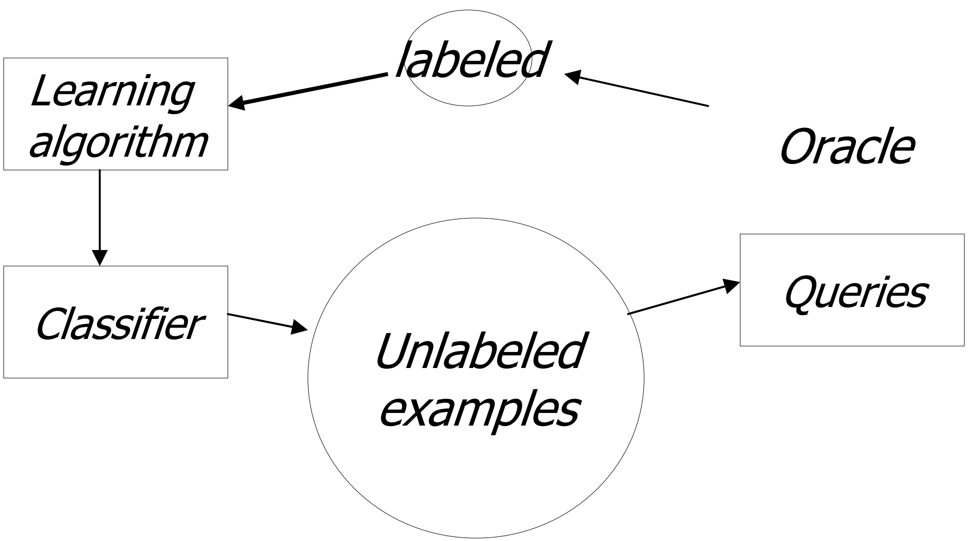
In some application problems:

- Limited number of labeled examples;
  Unlabeled examples are easily available
- Labeling costly
- Examples:
  - Classification of Web pages, email filtering, text categorization.
- Aims
  - An efficient classifier with a minimal number of additional labeling

David Cohn, Les Atlas, Richard Ladner - *Improving Generalization with Active Learning*, Machine Learning, 1994.

Active Learning

- Passive vs. Active Learning:
  - An algorithm controls input examples
- It is able to query (oracle / teacher) and receives a response (label) before outputting a final classifier
- How to select queries?



Previous Research on Active Learning

- Selective sampling [Cohn et al. 94]
- Uncertainty sampling [Lewis,Catlett 94]
- …
- Ensembles
  - Query by Committee of Two [Sueng et al.; Freund et al. 97]
  - Sampling committees
  - Query by Committee [Abe, Mamitsuka 98]
  - QBC and Active Decorate [Melvile, Mooney 04]

Query by Committee

L - set of labeled examples

U - set of unlabeled examples

A - base learning agorithm

k - number of act iterations

m - size of each sample

Repeat *k* times

1. Generate a committee of classifiers *C* = EnsembleMethod(A, L)*
2. For each *x* in *U* compute Info_val*(C*,x)*, based on the current committee
3. Select a subset *S* of *m* examples with max Info_val
4. Obtain Labels from Oracle for examples in *S*
5. Remove examples in *S* from *U* and add to *L*

Return *Ensemble*

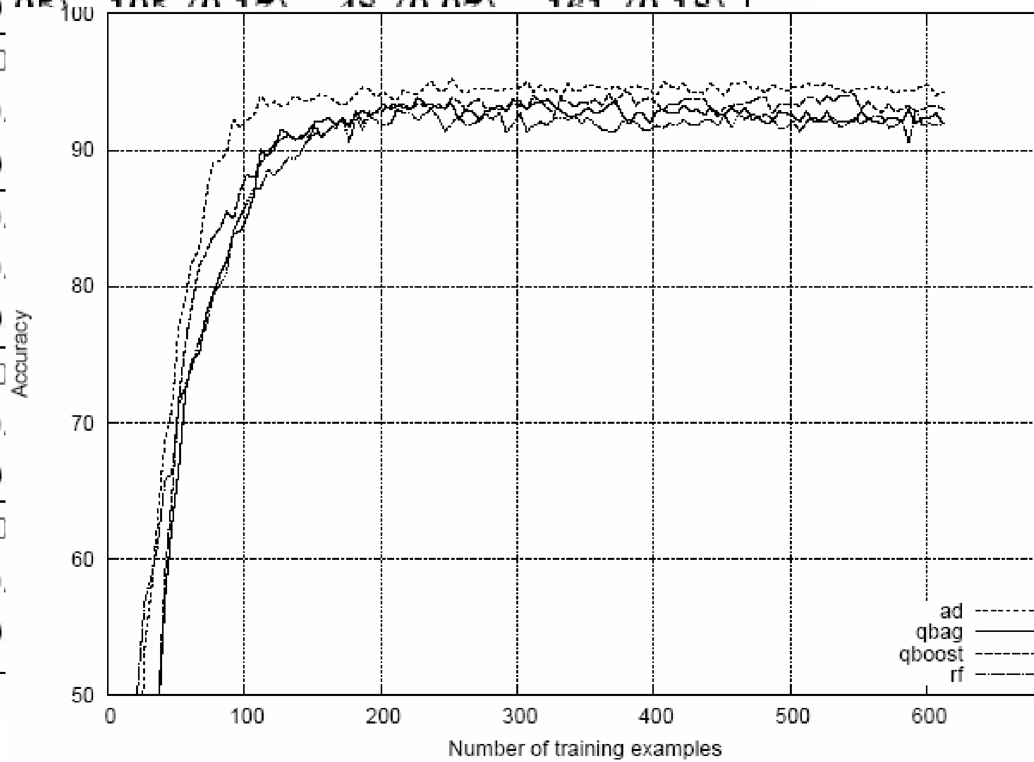
Remarks: *Info_val – disagreement measures, e.g. margins of classifiers*

Some experimental resuts

- Good classification accuracy + reduction of examples to be labeled.

TABLE 2: Reduction of the number of training examples to achieve the target accuracy - for Random forests results are presented for 50 trees and for 15.

Approach	wine	ionosphere	breast	soybean	diabetes	credit-g
Decorate	37 (0.23)	56 (0.18)	35 (0.06)	347 (0.57)	172 (0.25)	174 (0.19)
AD	27 (0.17)	36 (0.11)	37 (0.06)	125 (0.20)	410 (0.59)	267 (0.30)
AD k-nn	**22 (0.14)**	**32 (0.10)**	**30 (0** [illegible]	[illegible]	[illegible]	[illegible]
Bagging	122 (0.76)	206 (0.65)	388 (0 [illegible]	[illegible]	[illegible]	[illegible]
QBag	52 (0.33)	**46 (0.15)**	53 (0. [illegible]	[illegible]	[illegible]	[illegible]
QBag k-nn	**40 (0.25)**	48 (0.15)	**53 (0** [illegible]	[illegible]	[illegible]	[illegible]
Boosting	111 (0.69)	162 (0.51)	60 (0. [illegible]	[illegible]	[illegible]	[illegible]
QBoost	**75 (0.47)**	**102 (0.32)**	62 (0. [illegible]	[illegible]	[illegible]	[illegible]
Qboost k-nn	103 (0.64)	130 (0.41)	**54 (0** [illegible]	[illegible]	[illegible]	[illegible]
RF (50)	158 (0.99)	167 (0.53)	105 (0 [illegible]	[illegible]	[illegible]	[illegible]
ARF (50)	**56 (0.35)**	71 (0.23)	56 (0. [illegible]	[illegible]	[illegible]	[illegible]
ARF k-nn	128 (0.80)	**67 (0.21)**	**48 (0** [illegible]	[illegible]	[illegible]	[illegible]
RF (15)	115 (0.72)	251 (0.80)	105 (0 [illegible]	[illegible]	[illegible]	[illegible]
ARF (15)	**58 (0.36)**	118 (0.37)	65 (0. [illegible]	[illegible]	[illegible]	[illegible]
ARF k-nn	63 (0.39)	**86 (0.27)**	**48 (0** [illegible]	[illegible]	[illegible]	[illegible]



Learning from Changing Environments

Handling Concept Drift

Introduction

- Processing of data streams is considered
  - Continuous-incremental, ordered, huge …, data
  - Changing vs. statistic environments
  - Many applications generate streams of data
    - Sensor networks, monitoring telecomunication systems, traffic managements, classification of news, documents, ubiquitous environments, etc.
- Evolving classification data $\rightarrow$ concept drift
  - A target class definition changes over time
  - A classifier trained on the currently available data may fail if data distributions change
- New requirements for learning algorithms:
  - Handling and adapting to concept drift in data streams

Few previous research efforts

Older machine learning or AI directions

- Incremental learning vs. batch
  - Neural networks
  - Generalizations of k-NN (Aha’s IBL)
  - Bayesian update
- Incremental versions of symbolic knowledge reconstruction
  - Decision trees ID5 (Utgoff)
  - Clustering – COBWEB
- Another heuristic evaluation measures
- Specific sampling for larger data

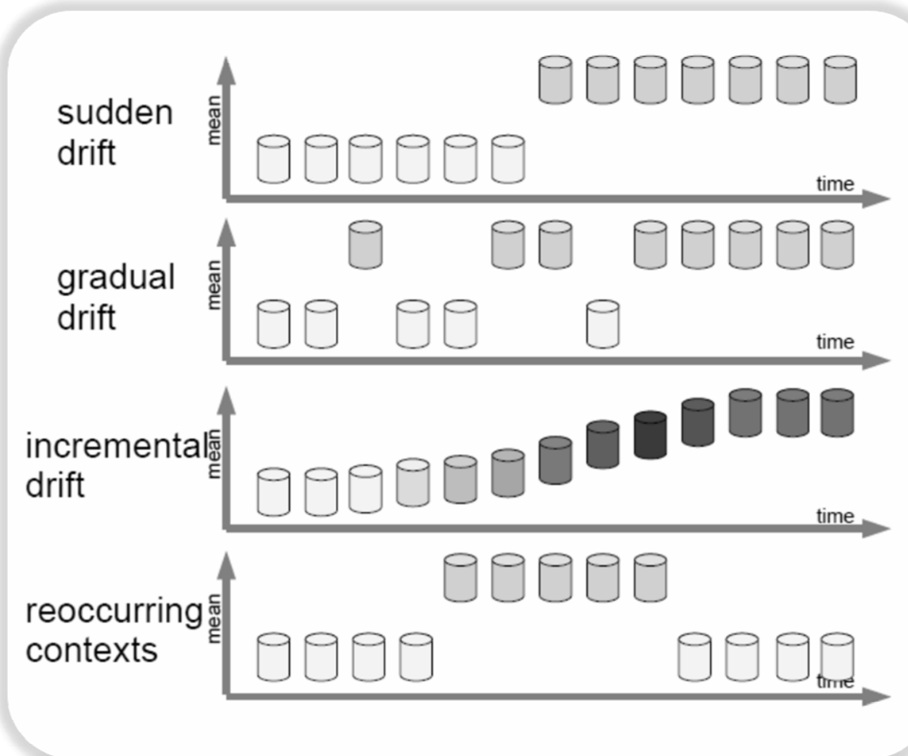
Tradional vs. Stream Processing

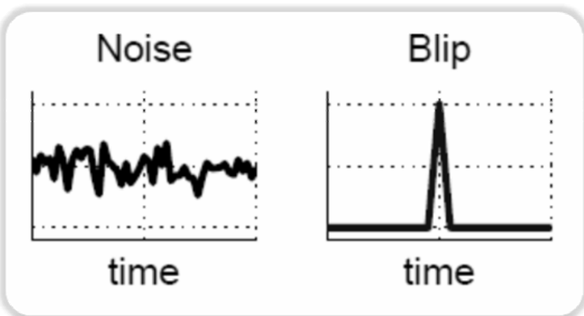
	Traditional	Stream
No. of passes	Muliple	Single
Processing Time	Unlimited	Restricted
Memory Usage	Unlimited	Restricted
Type of Results	Accurate	Approximate
Distributed	No	Yes

Concept drift

- Definitions of target classes change over time
  - Hidden context [Widmer,Kubat]

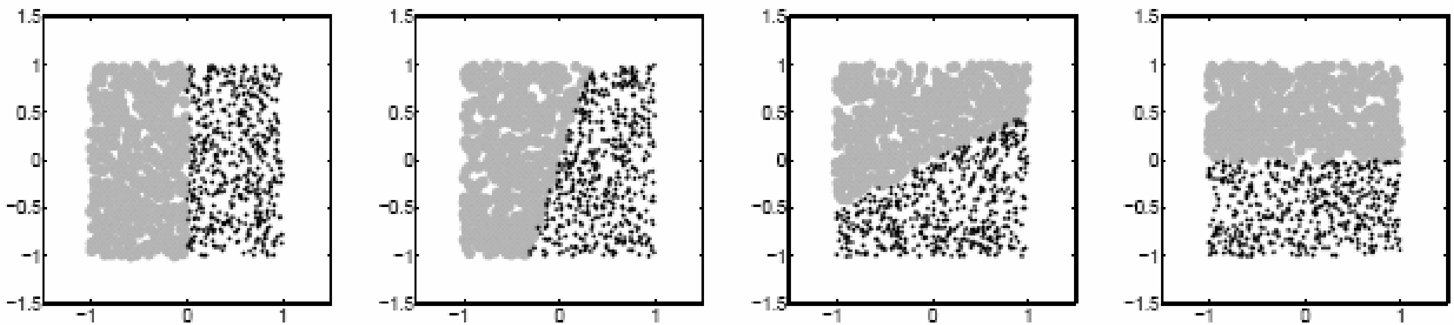
- Types of concept changes
  - Sudden (abrupt) concept drift
    - New classes appear, older not present $C_i(x) \neq C_t(x)$
  - Incremental, gradual drift
    - Slower distribution changes:
  - Recurrent concepts
- Do not react to noise, etc.





Gradual concept drift

- Rotating decision boundary problem



$v_1(t) = 1, v_2(t) = 0$
$v_3(t) = 0, v_4(t) = 0$
(a)

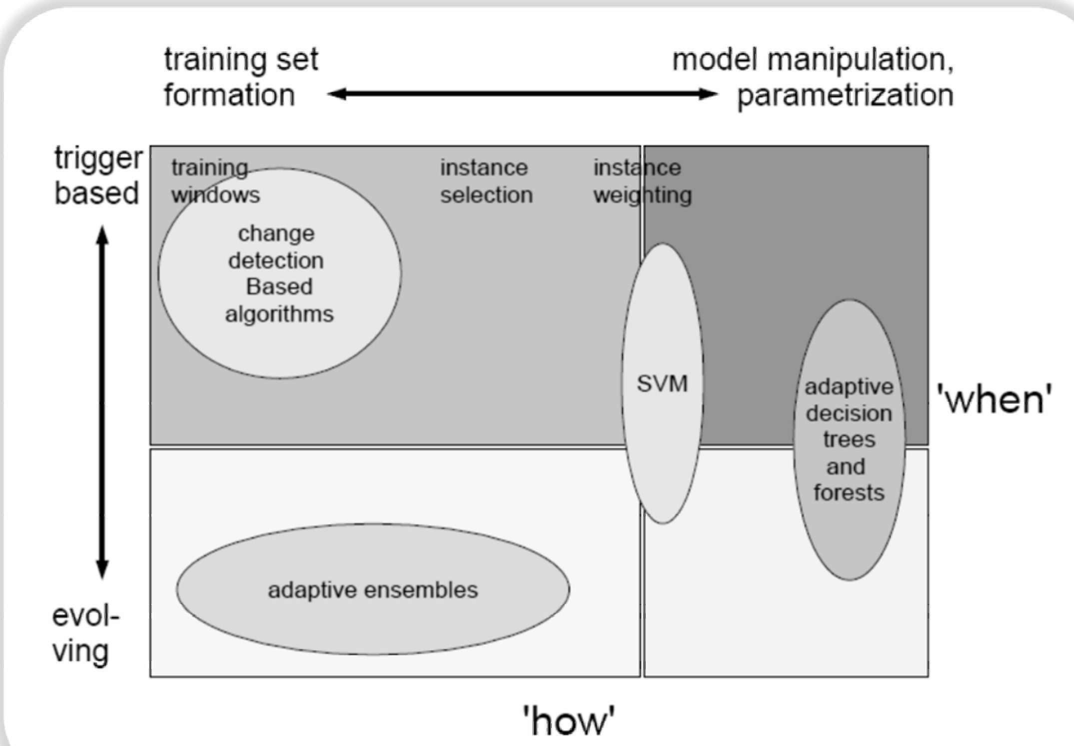
$v_1(t) = 0, v_2(t) = 1$
$v_3(t) = 0, v_4(t) = 0$
(b)

$v_1(t) = 0, v_2(t) = 0$
$v_3(t) = 1, v_4(t) = 0$
(c)

$v_1(t) = 0, v_2(t) = 0$
$v_3(t) = 0, v_4(t) = 1$
(d)

Methods for Learning Under Concept Drift

- Triggers → Detect and Retrain, e.g. DDT [Gama]
- Evolving → Constant Updates
  - Use a moving window with the latest N examples / fixed or variable size.
  - Use data chuncks
    - Special ensembles



Inductive Logic Programming - ILP

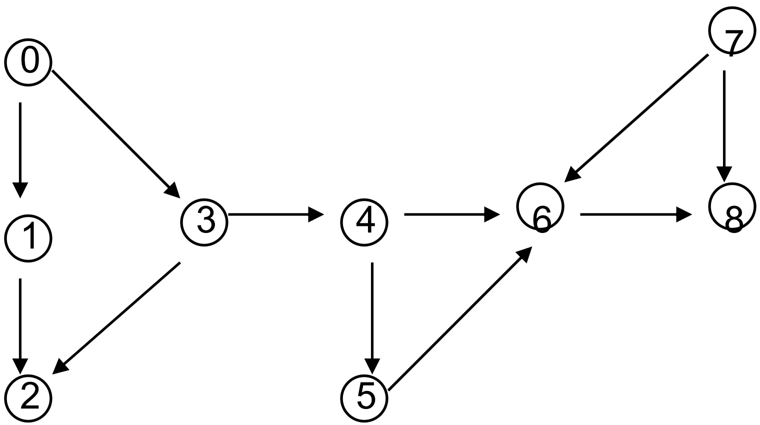
See also the longer lecture at my Web page [in Polish]

Learning First Order Rules

- Is object/attribute table sufficient data representation?
- Some limitations:
  - Representation expressivness – unable to express relations between objects or object elements. ,
  - *background knowledge* sometimes is quite complicated.
- Can learn sets of rules such as
  - $Parent(x,y) \rightarrow Ancestor(x,y)$
  - $Parent(x,z)$ and $Ancestor(z,y) \rightarrow Ancestor(x,y)$
- Research field of **Inductive Logic Programming**.

Why ILP? (slide of S.Matwin)

- **expressiveness of logic as representation** (Quinlan)



- can’t represent this graph as a fixed length vector of attributes
- can’t represent a “transition” rule:

A can-reach B if A link C, and C can-reach B

without variables

FINITE ELEMENT MESH DESIGN

Given a geometric **structure** and **loadings/boundary conditions**
Find an **appropriate resolution** for a finite element mesh

Examples: ten structures with appropriate meshes (cca. 650 edges)

Background knowledge

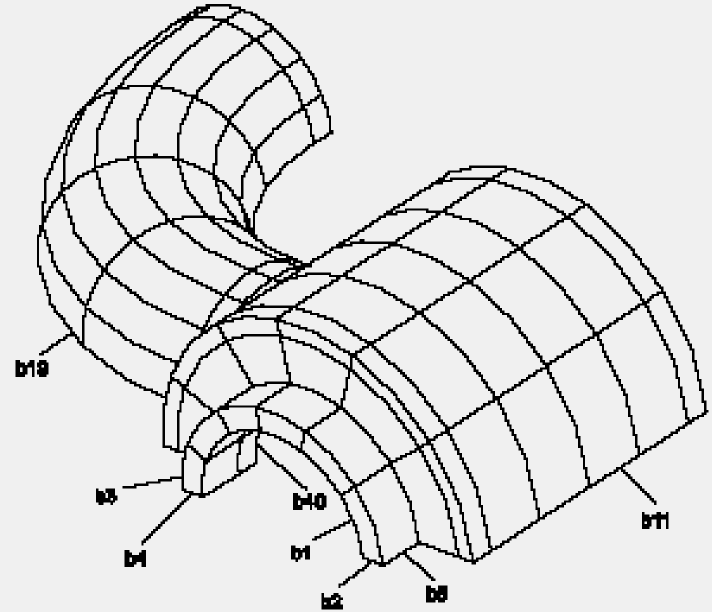
- Properties of edges (short, loaded, two-side-fixed, ...)
- Relations between edges (neighbor, opposite, equal)

ILP systems applied: GOLEM, CLAUDIEN

Many interesting rules discovered (according to expert evaluation)

Finite element mesh design (ctd.)

Example structure with an appropriate mesh



Example rules

$$mesh(Edge, 7) \leftarrow usual\_length(Edge),$$
$$neighbour\_xy(Edge, EdgeY), two\_side\_fixed(EdgeY),$$
$$neighbour\_zx(EdgeZ, Edge), not\_loaded(EdgeZ)$$
$$mesh(Edge, N) \leftarrow equal(Edge, Edge2), mesh(Edge2, N)$$

PAC model - Leslie Valiant

article | discussion | edit this page | history

Probably approximately correct learning

From Wikipedia, the free encyclopedia

In computational learning theory, **probably approximately correct learning (PAC learning)** is a framework for mathematical analysis of machine learning. It was proposed in 1984 by Leslie Valiant.[1]

In this framework, the learner receives samples and must select a generalization function (called the *hypothesis*) from a certain class of possible functions. The goal is that, with high probability (the "probably" part), the selected function will have low generalization error (the "approximately correct" part). The learner must be able to learn the concept given any arbitrary approximation ratio, probability of success, or distribution of the samples.

The model was later extended to treat noise (misclassified samples).

An important innovation of the PAC framework is the introduction of computational complexity theory concepts to machine learning. In particular, the learner is expected to find efficient functions (time and space requirements bounded to a polynomial of the example size), and the learner itself must implement an efficient procedure (requiring an example count bounded to a polynomial of the concept size, modified by the approximation and likelihood bounds).

Definitions and terminology [edit]

In order to give the definition for something that is PAC-learnable, we first have to introduce some terminology.[2]

For the following definitions, two examples will be used. The first is the problem of character recognition given an array of n bits. The other example is the problem of finding an interval that will correctly classify points within the interval as positive and the points outside of the range as negative.

Let X be a set call the *instance space* or the encoding of all the samples. In the character recognition problem, the instance space is $X = \{0,1\}^n$. In the interval problem the instance space is $X = \mathbb{R}$, where $\mathbb{R}$ denotes the set of all real numbers.

A *concept* is a subset $c \subset X$. One concept is the set of all of the bits that encode for the letter "P" in $X = \{0,1\}^n$. An example concept from the second example is the set of all of the numbers between $\pi/2$ and $\sqrt{10}$. A *concept class* is a set of concepts over X. This could be the set of all of the array of bits that are skeletonized 4-connected (width of the font is 1).

Let $EX(c,D)$ be a procedure draws an example, x, using a probability distribution D and gives the correct label $c(x)$.

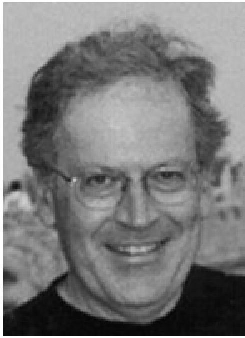
Say that there is an algorithm A that given access to $EX(c,D)$ and inputs ε and δ that, with probability at least $1 - \delta$, A outputs a hypothesis $h \in C$ that has error less than or equal to ε with examples drawn from X with the distribution D. If there is such an algorithm for every concept $c \in C$, for every distribution D over X, and for all $0 < \varepsilon < 1/2$ and $0 < \delta < 1/2$ then C is **PAC learnable**. We can also say that A is a **PAC learning algorithm** for C.

References [edit]

1. ^ L. Valiant. *A theory of the learnable.* Communications of the ACM, 27, 1984.
2. ^ Kearns and Vazirani, pg. 1-12.

Further reading [edit]

- M. Kearns, U. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994. A textbook.



za Wiki…

Any questions, remarks?

?

Thank you for your attention

Questions and remarks, please!

Contact, remarks:
Jerzy.Stefanowski@cs.put.poznan.pl