دانشگاه آزاد اسلامی واحد تبریز

نام درس: داده کاوی

بخش: الگوریتم های گروهی

نام استاد: دکتر مسعود کارگر

Roadmap

- Ensemble Learning: General Ideas
- Bootstrap Sampling
- Bagging
- Boosting
- Ensemble classifiers/clustering
- Success Story of Ensemble method
- General Idea of Ensemble methods

Ensemble Learning

- Employ multiple learners and combine their predictions.
- Methods of combination:
  - Bagging, boosting, voting
  - Error-correcting output codes
  - Mixtures of experts
  - Stacked generalization
  - Cascading
  - …
- **Advantage:** improvement in predictive accuracy.
- **Disadvantage:** it is difficult to understand an ensemble of classifiers

Why Do Ensembles Work?

Dietterich(2002) showed that ensembles overcome three problems:

- ***The Statistical Problem*** arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!
- ***The Computational Problem*** arises when the learning algorithm cannot guarantee finding the best hypothesis.
- ***The Representational Problem*** arises when the hypothesis space does not contain any good approximation of the target class(es).

T.G. Dietterich, Ensemble Learning, 2002.

Categories of Ensemble Learning

- Methods for Independently Constructing Ensembles
  - Bagging
  - Randomness Injection
  - Feature-Selection Ensembles
  - Error-Correcting Output Coding
- Methods for Coordinated Construction of Ensembles
  - Boosting
  - Stacking
  - Co-training

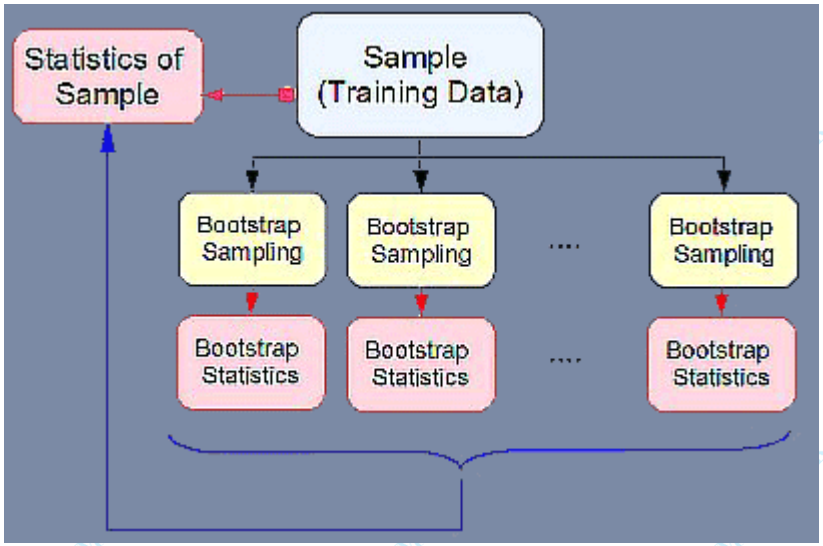
Methods for Independently Constructing Ensembles

One way to force a learning algorithm to construct multiple hypotheses is to **run the algorithm several times** and provide it with somewhat different data in each run. This idea is used in the following methods:

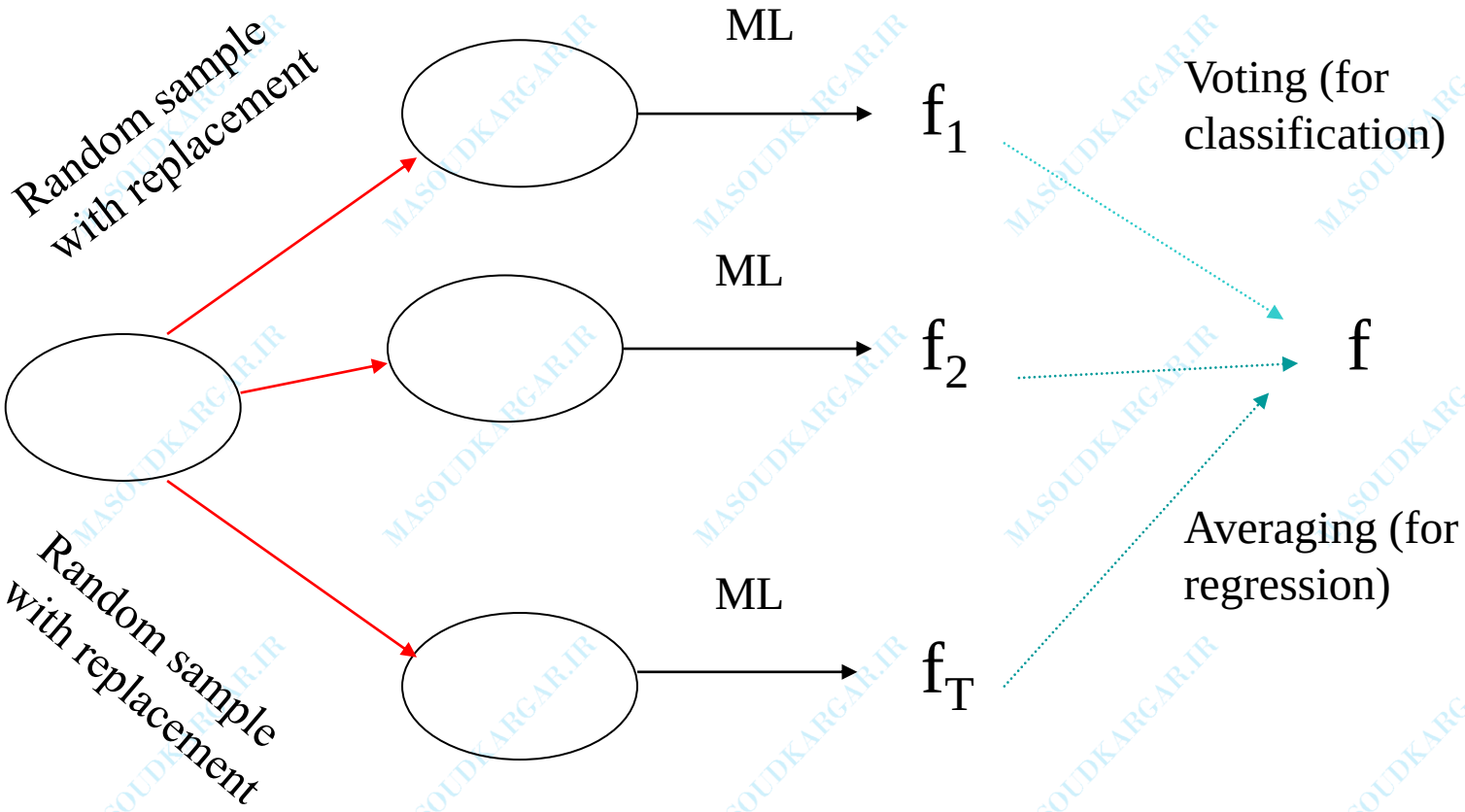
- ***Bagging***
- ***Randomness Injection***
- ***Feature-Selection Ensembles***
- ***Error-Correcting Output Coding.***

What is Bootstrap (sampling)?

- Bootstrap is sampling with replacement from a sample.
- The name may come from phrase “pull up by your own bootstraps” which mean ‘rely on your own resources



Bagging (Bootstrap Aggregration)



Bagging is only effective when using **unstable** (i.e. a small change in the training set can cause a significant change in the model) nonlinear models

Bias-variance Decomposition

- Theoretical tool for analyzing how much *specific* training set affects performance of a classifier
  - Total expected error: bias + variance
  - The ***bias*** of a classifier is the expected error of the classifier due to the fact that the classifier is not perfect
  - The ***variance*** of a classifier is the expected error due to the particular training set used

Why does bagging work?

- Bagging reduces variance by voting/ averaging, thus reducing the overall expected error
  - In the case of classification there are pathological situations where the overall error might increase
  - Usually, the more classifiers the better

Randomization Injection

- Inject some randomization into a standard learning algorithm (usually easy):
  - Neural network: random initial weights
  - Decision tree: when splitting, choose one of the top N attributes at random (uniformly)
- Dietterich (2000) showed that 200 randomized trees are statistically significantly better than C4.5 for over 33 datasets!

Random Forest algorithm

- If the number of cases in the training set is N, sample N cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.

- If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

- Each tree is grown to the largest extent possible. There is no pruning.

Features of Random Forests

- **It is unexcelled in accuracy among current algorithms.**
- It runs efficiently on large data bases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It has methods for balancing error in class population unbalanced data sets.
- Generated forests can be saved for future use on other data.
- Prototypes are computed that give information about the relation between the variables and the classification.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.

Feature-Selection Ensembles (Random Subspace Method)

- ***Key idea:*** Provide a different subset of the input features in each 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!

Error-correcting output codes

- Very elegant method of transforming multi-class problem into two-class problem
  - Simple scheme: as many binary class attributes as original classes using one-per-class coding

class	class vector
a	1000
b	0100
c	0010
d	0001

- Train f(ci) for each bit
- Idea: use *error-correcting codes* instead

Error-correcting output codes

- Example:

class	class vector
a	1111111
b	0000111
c	0011001
d	0101010

  - What's the true class if base classifiers predict 1011111?

Thomas G. Dietterich, Ghulum Bakiri. Journal of Articial IntelligenceResearch 2 1995. **Solving Multiclass Learning Problems via**. **Error-Correcting Output Codes.**

Hamming Distance as Quality measurement

Simple code

class	class vector
a	1000
b	0100
c	0010
d	0001

Dh=1

Error correction code

class	class vector
a	1111111
b	0000111
c	0011001
d	0101010

Dh=4

We use nearest code as the output code. So simple code no robustness.

Methods for Coordinated Construction of Ensembles

- ***The key idea*** is to learn **complementary** classifiers so that instance classification is realized by taking a weighted sum of the classifiers. This idea is used in following methods:
  - Boosting
  - Stacking.

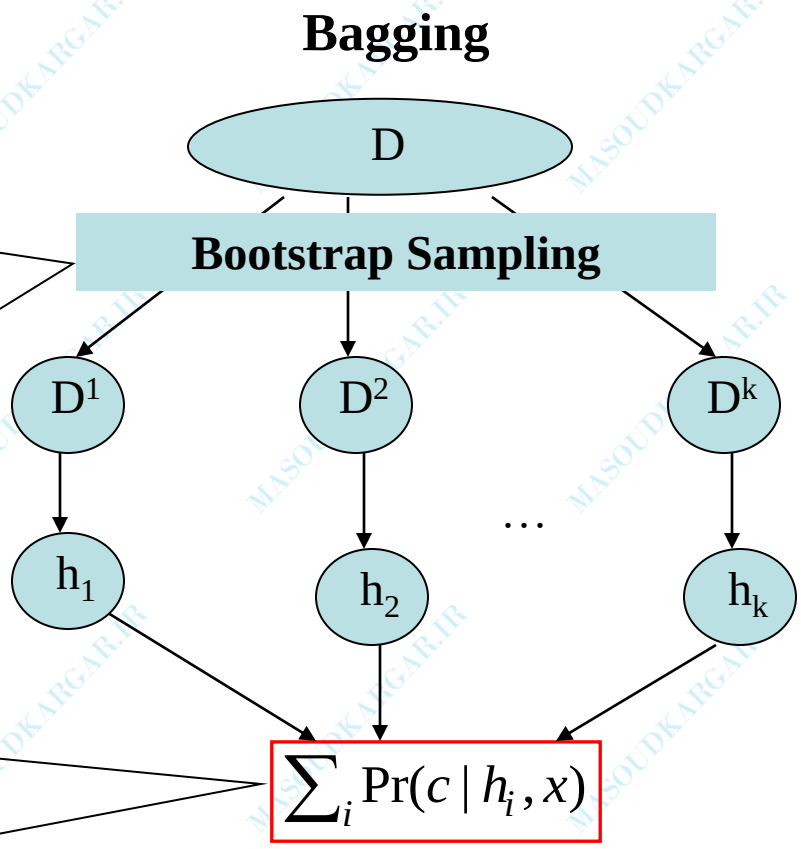
Inefficiency with Bagging

Inefficiency with bootstrap sampling:

- Every example has equal chance to be sampled
- No distinction between "easy" examples and "difficult" examples

Inefficiency with model combination

- A constant weight for each classifier
- No distinction between accurate classifiers and inaccurate classifiers



Improve the Efficiency of Bagging

- Better sampling strategy
  - Focus on the examples that are difficult to classify correctly
- Better combination strategy
  - Accurate model should be assigned with more weights

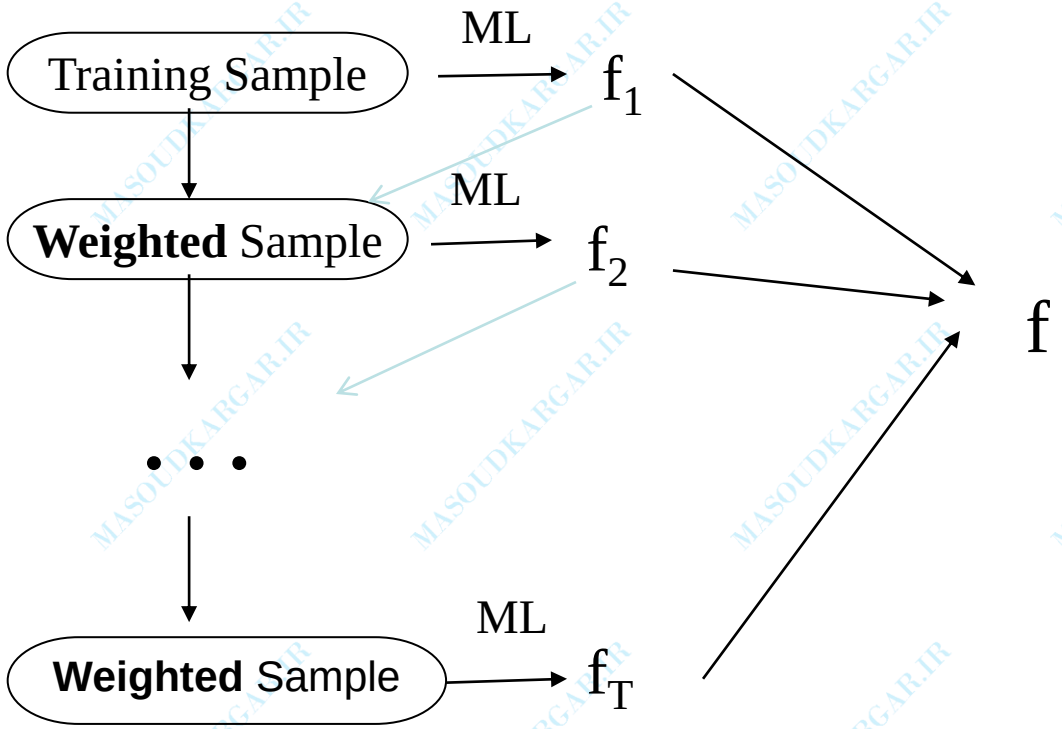
Overview of Boosting

- Introduced by Schapire and Freund in 1990s.
- “Boosting”: convert a weak learning algorithm into a strong one.
- Main idea: Combine many weak classifiers to produce a powerful committee.
- Algorithms:
  - **AdaBoost**: adaptive boosting
  - Gentle AdaBoost
  - BrownBoost
  - …

Boosting

- Also uses voting/averaging but models are weighted according to their performance
- Iterative procedure: new models are influenced by performance of previously built ones
  - New model is encouraged to become expert for instances classified incorrectly by earlier models
  - Intuitive justification: models should be experts that complement each other
- There are several variants of this algorithm

Boosting



Boosting: Use the same sample with different weights to generate classifiers
Bagging: Use different samples with identical weights to generate classifiers

Intuition of Boosting

- Train a set of weak hypotheses: h_1, …., h_T.
- The combined hypothesis H is a **weighted** majority vote of the T weak hypotheses.
  ➔Each hypothesis h_t has a weight α_t.

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

- During the training, focus on the examples that are misclassified.
  ➔ At round t, example x_i has the weight $D_t(i)$.

Basic Setting

- Binary classification problem
- Training data:

$$(x_1, y_1), \ldots, (x_m, y_m), where \ x_i \in X, y_i \in Y = \{-1, 1\}$$

- $D_t(i)$: the weight of x_i at round t. $D_1(i)=1/m$.
- A learner L that finds a weak hypothesis h_t: X ➔ Y given the training set and D_t
- The error of a weak hypothesis h_t:

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i).$$

The Basic AdaBoost Algorithm

For t=1, …, T

- Train weak learner using training data and D_t
- Get ht: X ➔ {-1,1} with error

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$$

- Choose

$$\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

- Update

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} * \begin{cases} e^{-\alpha_t} & if\ h_t(x_i) = y_i \\ e^{\alpha_t} & if\ h_t(x_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

The general AdaBoost algorithm

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

AdaBoost Example

Train data

x1	x2	y
1	5	+
2	3	+
3	2	–
4	6	–
4	7	+
5	9	+
6	5	–
6	7	+
8	5	–
8	8	–

D1
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
0.10
1.00

Initialization

Round 1

h1e	ε	D2
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
0	0.00	0.07
1	0.10	0.17
1	0.10	0.17
0	0.00	0.07
1	0.10	0.17
0	0.00	0.07
0	0.00	0.07
ε1	0.30	1.00
α1	0.42	↕
	Zt	0.92

Round 2

h2e	ε	D3
0	0.00	0.05
0	0.00	0.05
1	0.07	0.17
1	0.07	0.17
0	0.00	0.11
0	0.00	0.11
1	0.07	0.17
0	0.00	0.11
0	0.00	0.05
0	0.00	0.05
ε2	0.21	1.00
α2	0.65	↕
	Zt	0.82

Round 3

h3e	ε
1	0.05
1	0.05
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
0	0.00
1	0.05
ε3	0.14
α3	0.92

Multiclass problems

- There are k possible classes.

- Approaches:
  - AdaBoost.MH
  - AdaBoost.MI

AdaBoost.MH

- Training time: (K class)
  - Train one classifier: f(x'), where x'=(x,c)
  - Replace (x,y) with k derived examples
    - ((x,1), 0)
    - …
    - ((x, y), 1)
    - …
    - ((x, k), 0)
- Decoding time: given a new example x
  - Run the classifier f(x, c) on k derived examples:
    (x, 1), (x, 2), …, (x, k)
  - Choose the class c with the highest confidence score f(x, c).

AdaBoost.MI

- Training time:
  - Train k independent classifiers: $f_1(x)$, $f_2(x)$, …, $f_k(x)$
  - When training the classifier f_c for class c, replace (x,y) with
    - (x, 1) if y = c
    - (x, 0) if y != c
- Decoding time: given a new example x
  - Run each of the k classifiers on x
  - Choose the class with the highest confidence score $f_c(x)$.

Remarks on Boosting

- Boosting can be applied without weights using re-sampling with probability determined by weights;
- Boosting decreases exponentially the training error in the number of iterations;
- Boosting works well if base classifiers are not too complex and their error doesn't become too large too quickly!

Strengths of AdaBoost

- It has no parameters to tune (except for the number of rounds)
- It is fast, simple and easy to program (??)
- It comes with a set of theoretical guarantee (e.g., training error, test error)
- Instead of trying to design a learning algorithm that is accurate over the entire space, we can focus on finding base learning algorithms that only need to be better than random.
- It can identify outliners: i.e. examples that are either mislabeled or that are inherently ambiguous and hard to categorize.

Weakness of AdaBoost

- The actual performance of boosting depends on the data and the base learner.
- Boosting seems to be especially susceptible to **noise**.
- When the number of outliners is very large, the emphasis placed on the hard examples can hurt the performance.

  ➔ “Gentle AdaBoost”, “BrownBoost”

Comparison of Bagging and Boosting

- Bagging always uses resampling rather than reweighting.
- Bagging does not modify the distribution over examples or mislabels, but instead always uses the uniform distribution
- In forming the final hypothesis, bagging gives equal weight to each of the weak hypotheses

Sources of Errors:

- Bias and Variance
  - Bias arises when the classifier cannot represent the true function – that is, the classifier underfits the data
  - Variance arises when the classifier overfits the data
  - There is often a tradeoff between bias and variance

Effect of Bagging

- If the bootstrap replicate approximation were correct, then bagging would reduce variance without changing bias.

- In practice, bagging can reduce both bias and variance
  - For high-bias classifiers, it can reduce bias
  - For high-variance classifiers, it can reduce variance

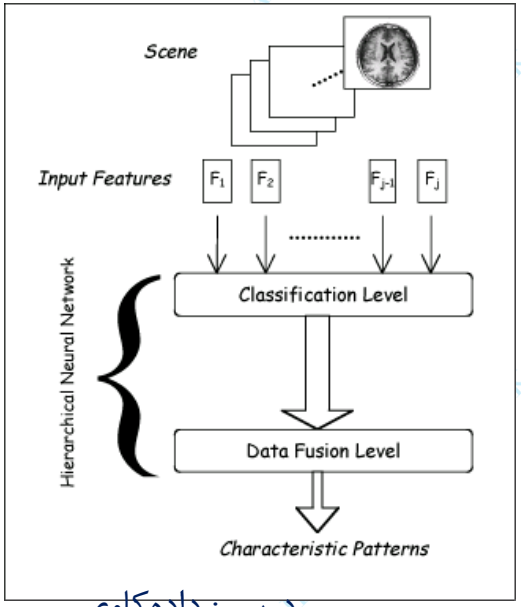
Effect of Boosting

- In the early iterations, boosting is primary a bias-reducing method
- In later iterations, it appears to be primarily a variance-reducing method

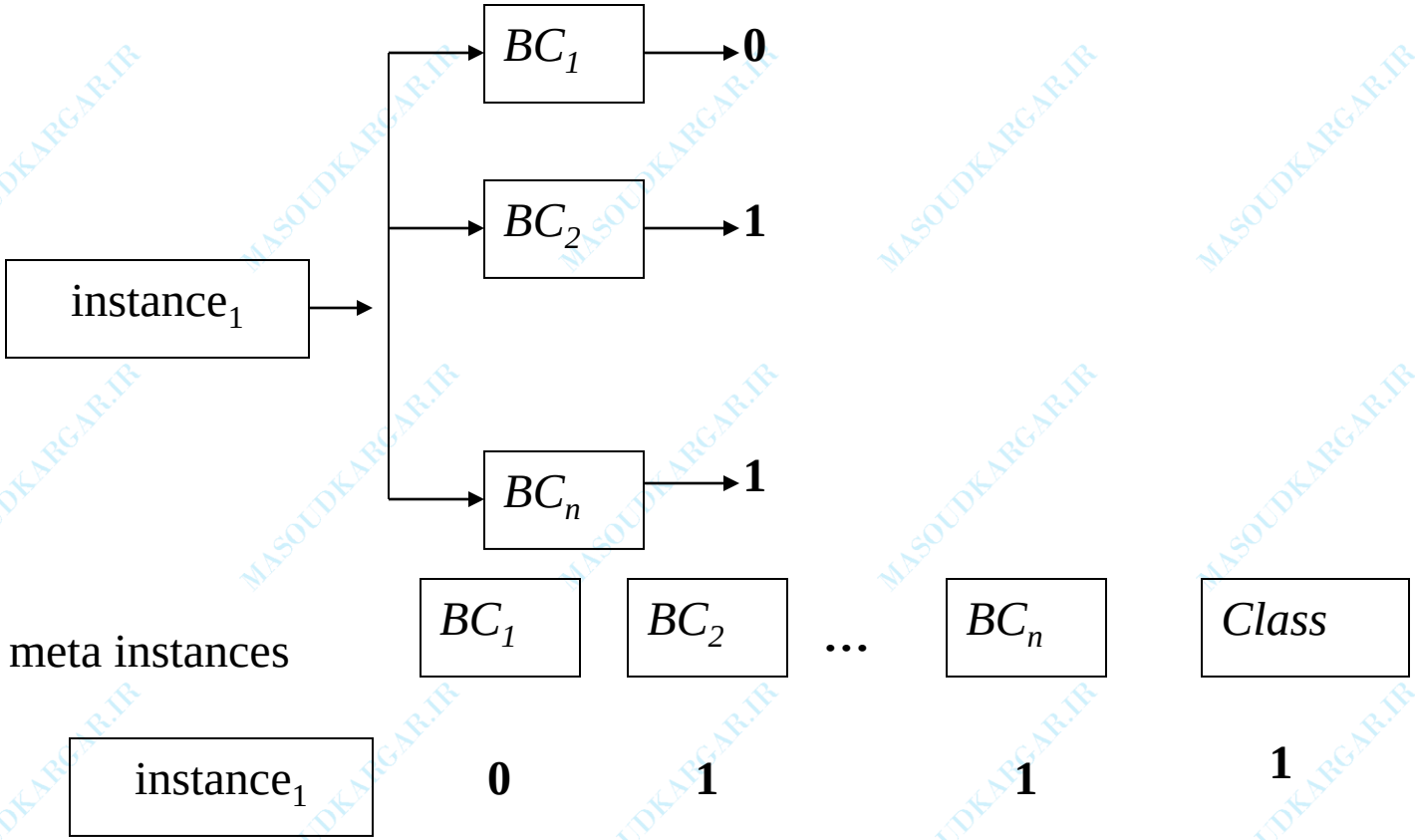
Stacking

- Uses *meta learner* instead of voting to combine predictions of base learners
  - Predictions of base learners (*level-0 models*) are used as input for meta learner (*level-1 model)*
- Base learners usually different learning schemes
- Hard to analyze theoretically: “black magic”

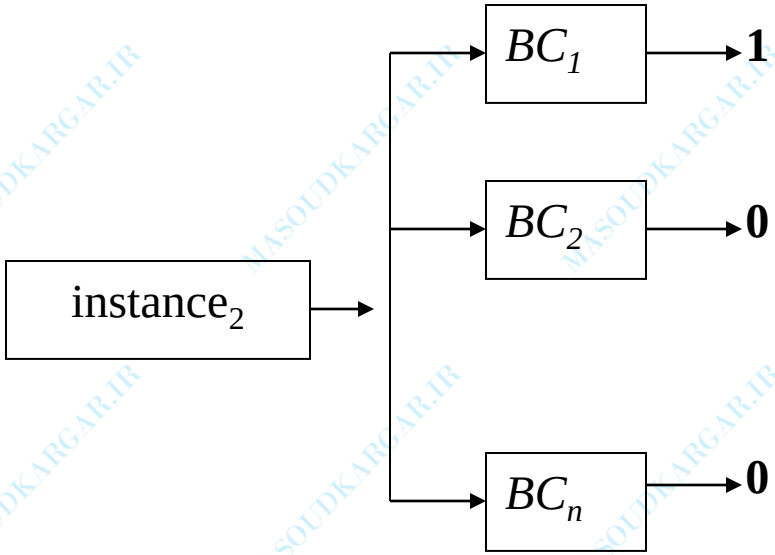
Hierarchical Neural Networks



Stacking


BC_1
0
BC_2
1
instance$_1$
BC_n
1
meta instances
BC_1
BC_2
...
BC_n
Class
instance$_1$
0
1
1
1

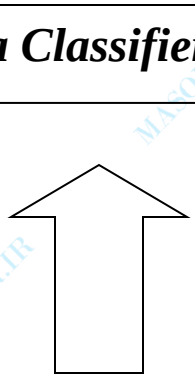
Stacking



meta instances

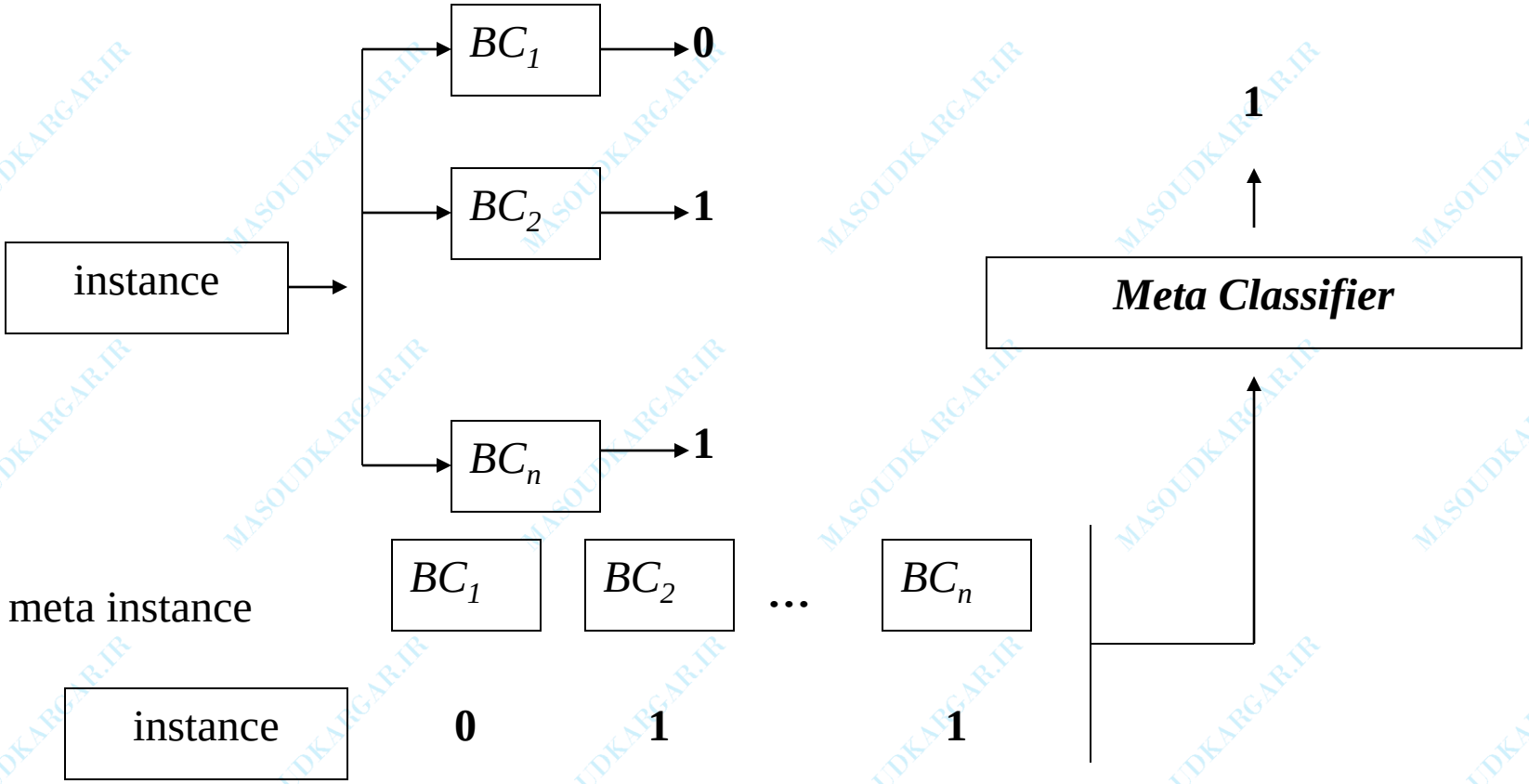
	BC_1	BC_2	…	BC_n	Class
instance$_1$	0	1		1	1
instance$_2$	1	0		0	0

Stacking



meta instances	BC_1	BC_2	...	BC_n	Class
$instance_1$	**0**	**1**		**1**	**1**
$instance_2$	**1**	**0**		**0**	**0**

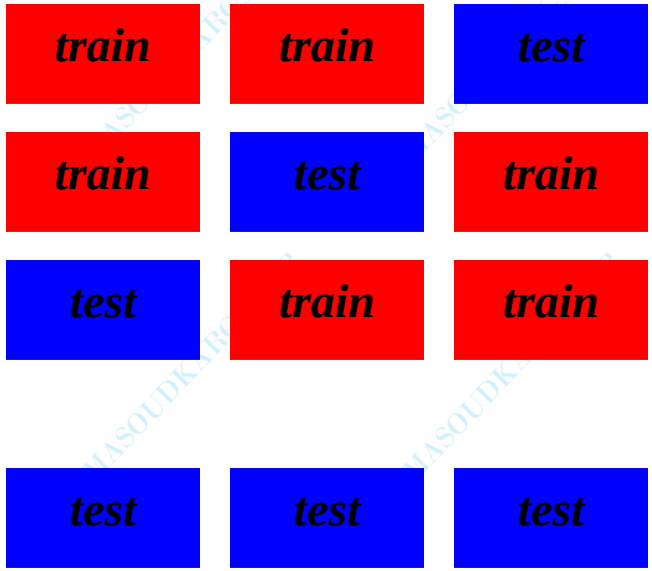
Stacking


BC_1
0
BC_2
1
instance
BC_n
1
1
Meta Classifier
meta instance
BC_1
BC_2
...
BC_n
instance
0
1
1

More on stacking

- Predictions on training data can’t be used to generate data for level-1 model! The reason is that the level-0 classifier that better fit training data will be chosen by the level-1 model! Thus,

- k-fold cross-validation-like scheme is employed! An example for k = 3!



More on stacking

- If base learners can output probabilities it's better to use those as input to meta learner
- Which algorithm to use to generate meta learner?
  - In principle, any learning scheme can be applied
  - David Wolpert: "relatively global, smooth" model
    - » Base learners do most of the work
    - » Reduces risk of overfitting

Some Practical Advices

- If the classifier is **unstable** (i.e, decision trees) then apply bagging!
- If the classifier is **stable and simple** (e.g. Naïve Bayes) then apply boosting!
- If the classifier is **stable and complex** (e.g. Neural Network) then apply randomization injection!
- If you have many classes and a binary classifier then try error-correcting codes! If it does not work then use a complex binary classifier!

Summary

- Boosting combines many weak classifiers to produce a powerful committee.
- It comes with a set of theoretical guarantee (e.g., training error, test error)
- It performs well on many tasks.
- It is related to many topics (TBL, MaxEnt, linear programming, etc)

Slides Credits

- Slides in this presentation is partially based on the work of
  - Rong Jin (Michigan State University)
  - Yaochu Jin (Future Technology Research
    Honda R&D Europe (Germany)
  - Evgueni Smirnov (www.cs.unimaas.nl/**smirnov)**

Special Appreciation

- Dr. Jianjun Hu
  http://mleg.cse.sc.edu/edu/csce822/
- University of South Carolina
- Department of Computer Science and Engineering