CS 564: Ensemble Learning

Asif Ekbal

Dept. of Computer Science and Engineering
IIT Patna, India-800 013
Email: asif@iitp.ac.in
asif.ekbal@gmail.com

Drawbacks of Single Classifier

- The “best” classifier not necessarily the ideal choice

- For solving a classification problem, many individual classifiers with different parameters are trained
  - The “best” classifier will be selected according to some criteria e.g., *training accuracy* or *complexity of the classifiers*

- Problems: Which one is the best?
  - Maybe more than one classifiers meet the criteria (e.g. same training accuracy), especially in the following situations:

    -Without sufficient training data

    - Learning algorithm leads to different local optimas easily

Drawbacks of Single Classifier

- Potentially valuable information may be lost by discarding the results of less-successful classifiers

  E.g., the discarded classifiers may correctly classify some samples

- Other drawbacks
  - Final decision must be wrong if the output of selected classifier is wrong
  - Trained classifier may not be complex enough to handle the problem

Ensemble Learning

- Employ multiple learners and combine their predictions

- Methods of combination:
  - Bagging, boosting, voting
  - Error-correcting output codes
  - 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:

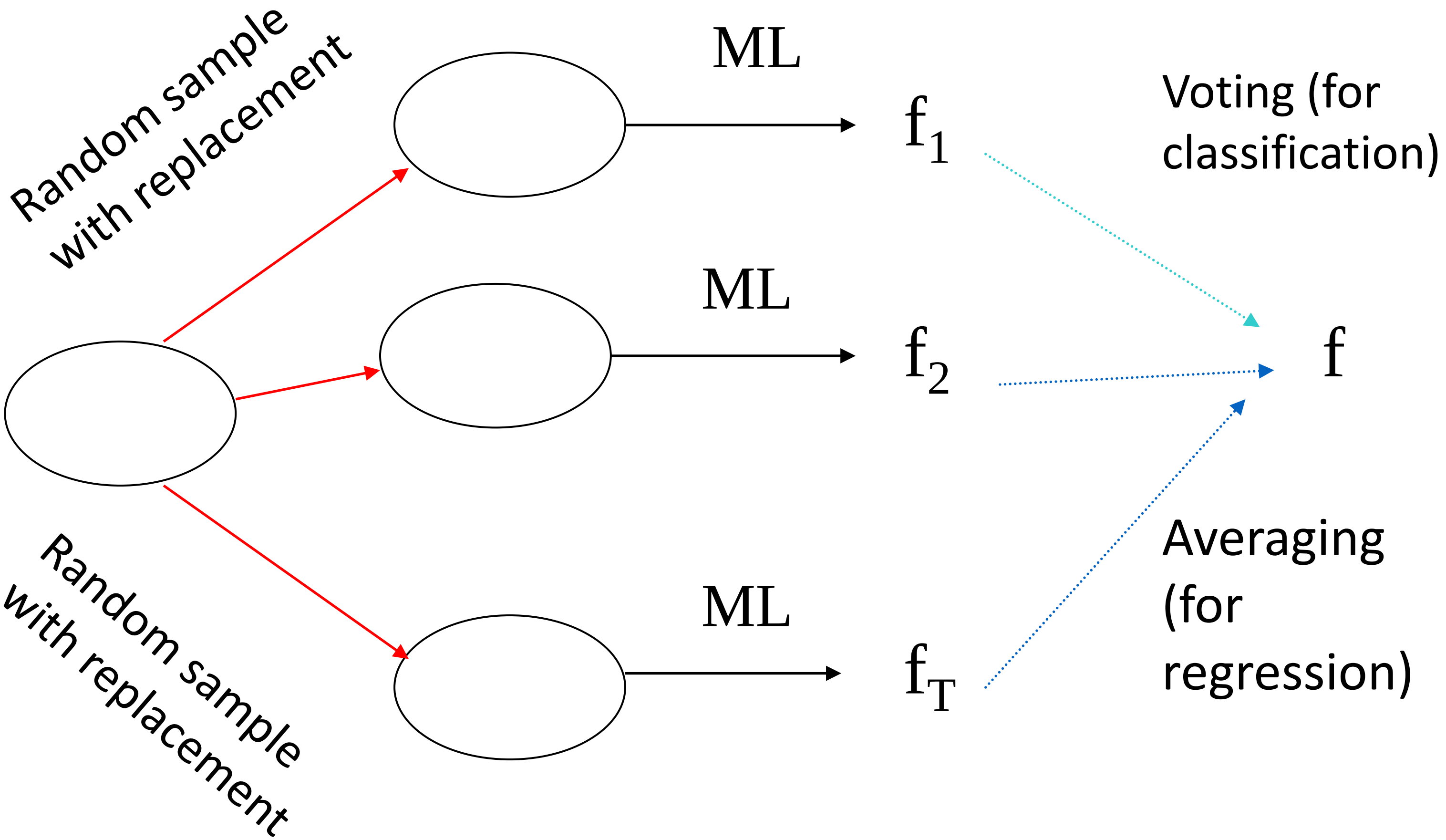
- ***Statistical Problem-*** arises when the hypothesis space is too large for the amount of available data. Hence, there are many hypotheses with the same accuracy on the data and the learning algorithm chooses only one of them! There is a risk that the accuracy of the chosen hypothesis is low on unseen data!
- ***Computational Problem-*** arises when the learning algorithm cannot guarantee finding the best hypothesis.
- ***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

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

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!

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?

ECOC-more.ppt

Dietterich, Ghulum Bakiri.

Journal of Artificial Intelligence Research 2 1995. Solving **Multiclass Learning Problems via**. **Error-Correcting Output Codes.**

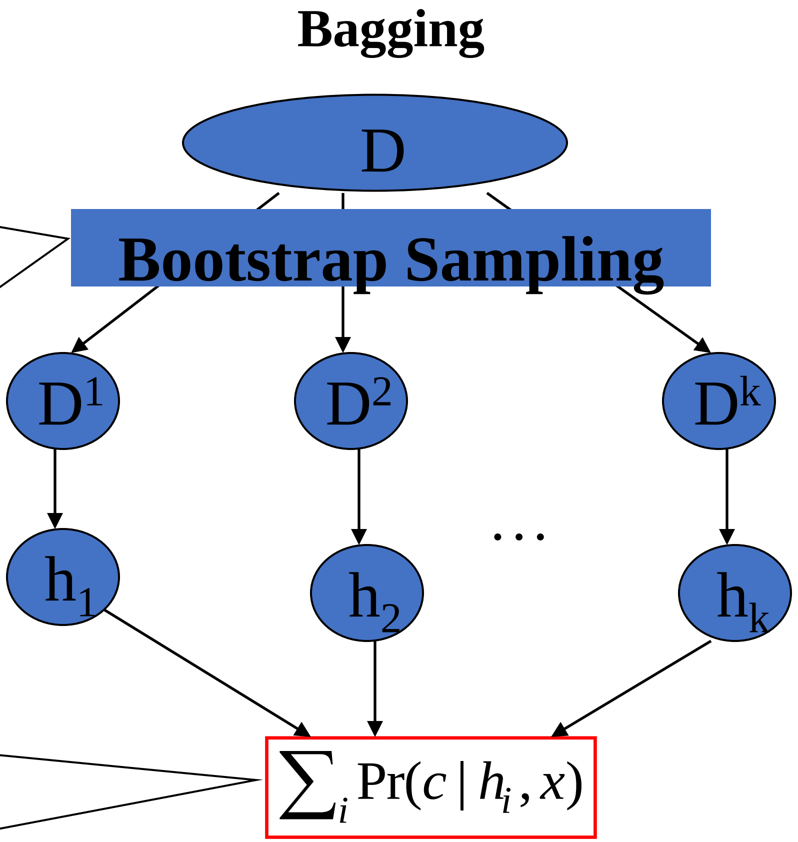
Methods for Coordinated Construction of Ensembles

- ***Key idea-*** to learn *complementary* classifiers so that instance classification is realized by taking a weighted sum of the classifiers:
  - 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

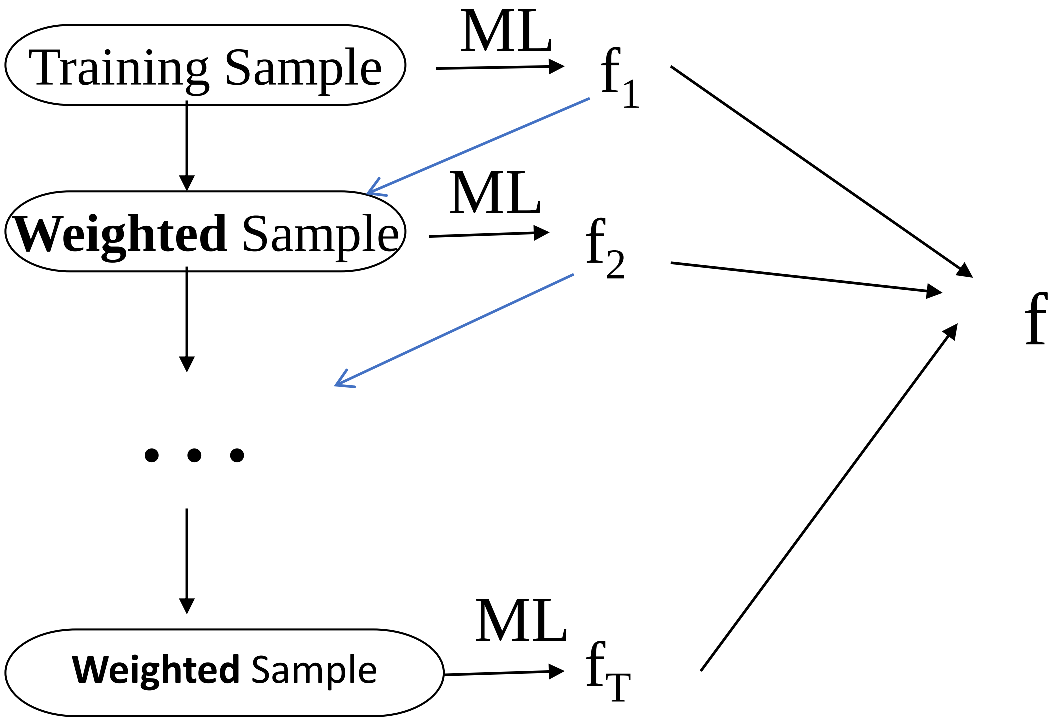
- Main idea: Combine many weak classifiers to produce a powerful committee

- Algorithms:
  - **AdaBoost**: adaptive boosting
  - Gentle AdaBoost
  - BrownBoost
  - …

Boosting

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

Boosting



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

Bagging: Use different samples with identical weights to generate classifiers

AdaBoost

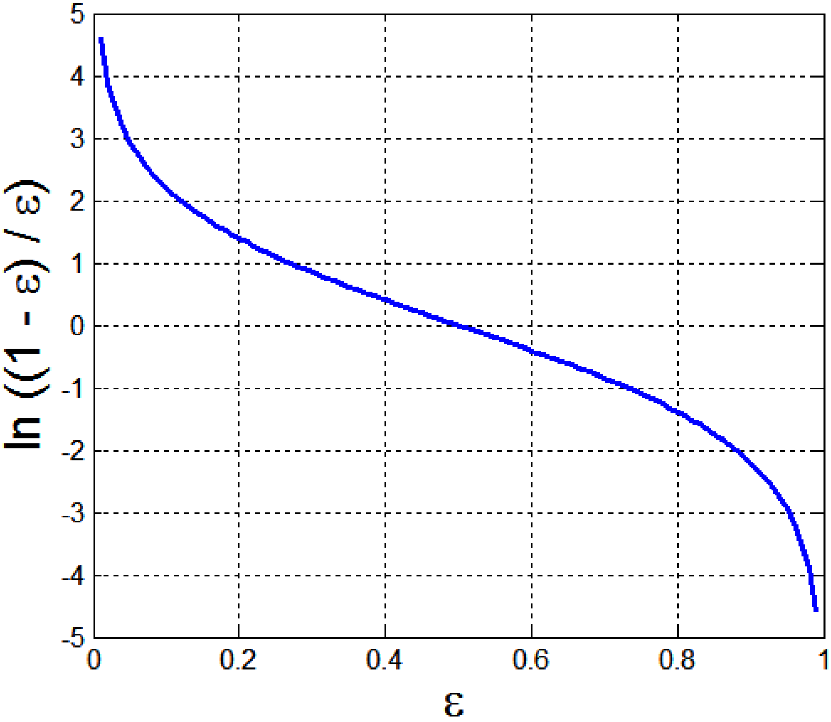
Base classifiers: C_1, C_2, …, C_T

Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$



AdaBoost Algorithm

Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

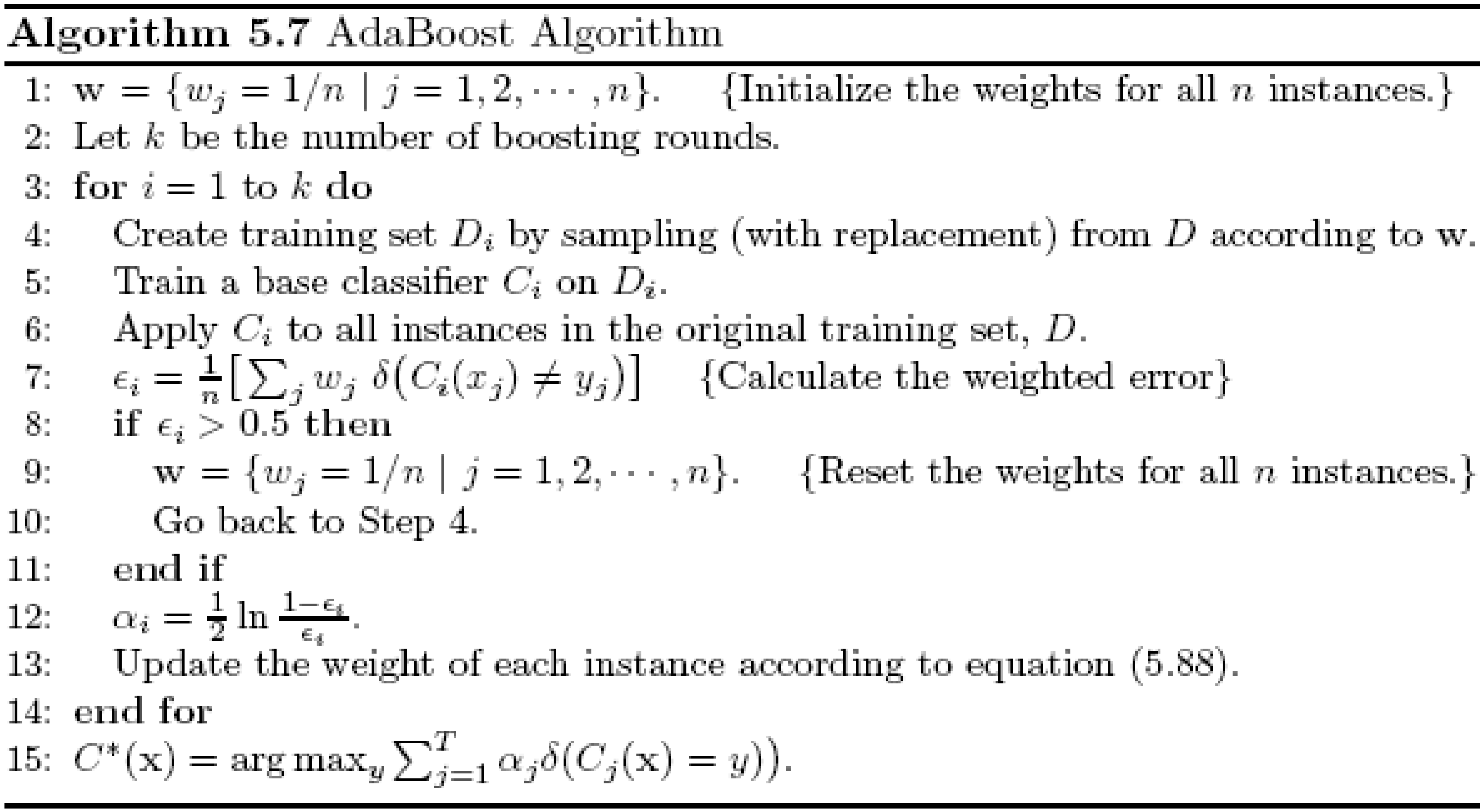
$$\text{where } Z_j \text{ is the normalization factor}$$

If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated

Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\big(C_j(x) = y\big)$$

AdaBoost Algorithm



Algorithm 5.7 AdaBoost Algorithm

```
 1: $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$.    {Initialize the weights for all $n$ instances.}
 2: Let $k$ be the number of boosting rounds.
 3: for $i = 1$ to $k$ do
 4:    Create training set $D_i$ by sampling (with replacement) from $D$ according to $\mathbf{w}$.
 5:    Train a base classifier $C_i$ on $D_i$.
 6:    Apply $C_i$ to all instances in the original training set, $D$.
 7:    $\epsilon_i = \frac{1}{n}\left[\sum_j w_j \, \delta\big(C_i(x_j) \neq y_j\big)\right]$    {Calculate the weighted error}
 8:    if $\epsilon_i > 0.5$ then
 9:       $\mathbf{w} = \{w_j = 1/n \mid j = 1, 2, \cdots, n\}$.    {Reset the weights for all $n$ instances.}
10:       Go back to Step 4.
11:    end if
12:    $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
13:    Update the weight of each instance according to equation (5.88).
14: end for
15: $C^*(\mathbf{x}) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y)\big)$.
```

Strengths of AdaBoost

- No parameters to tune (except for the number of rounds)
- Fast, simple and easy to program (??)
- Comes with a set of theoretical guarantee (e.g., *training error*, *test error etc*)

- 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

- Can identify outliers: i.e. examples that are either mislabeled or inherently ambiguous and hard to categorize

Weakness of AdaBoost

- Actual performance depends on the data and the base learner

- Boosting seems to be especially susceptible to **noise**

- When the number of outliers 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 *re-sampling* rather than *re-weighting*

- 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

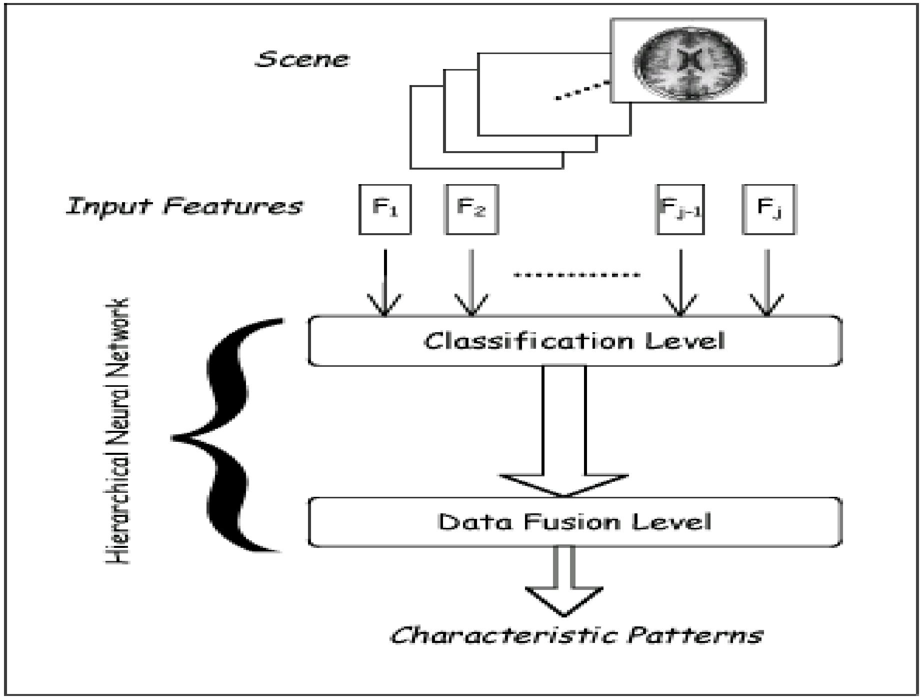
Sources of Errors

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

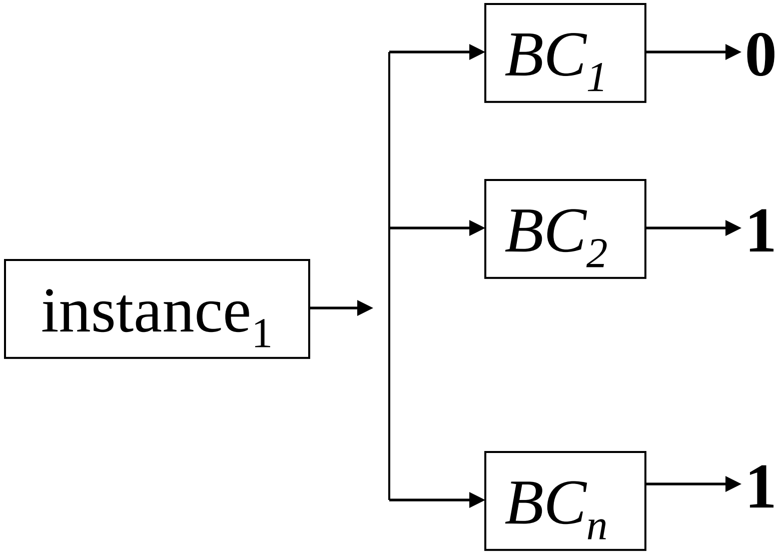
Stacking

- 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

Hierarchical Neural Networks

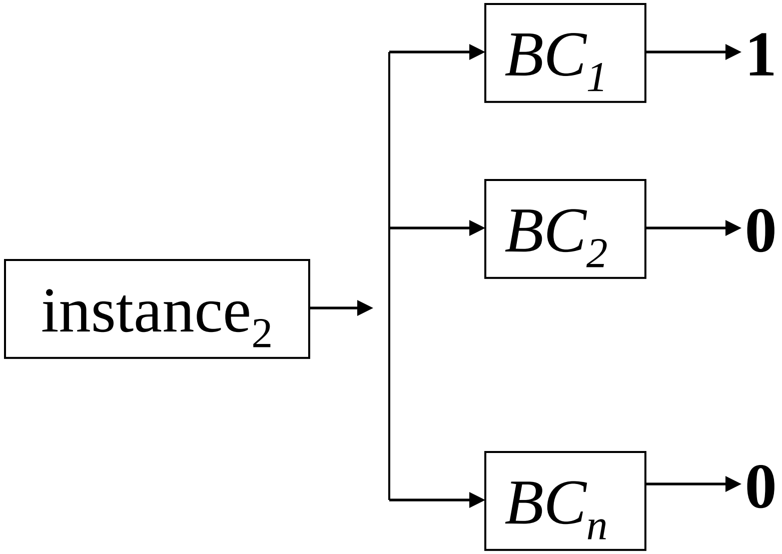


Stacking



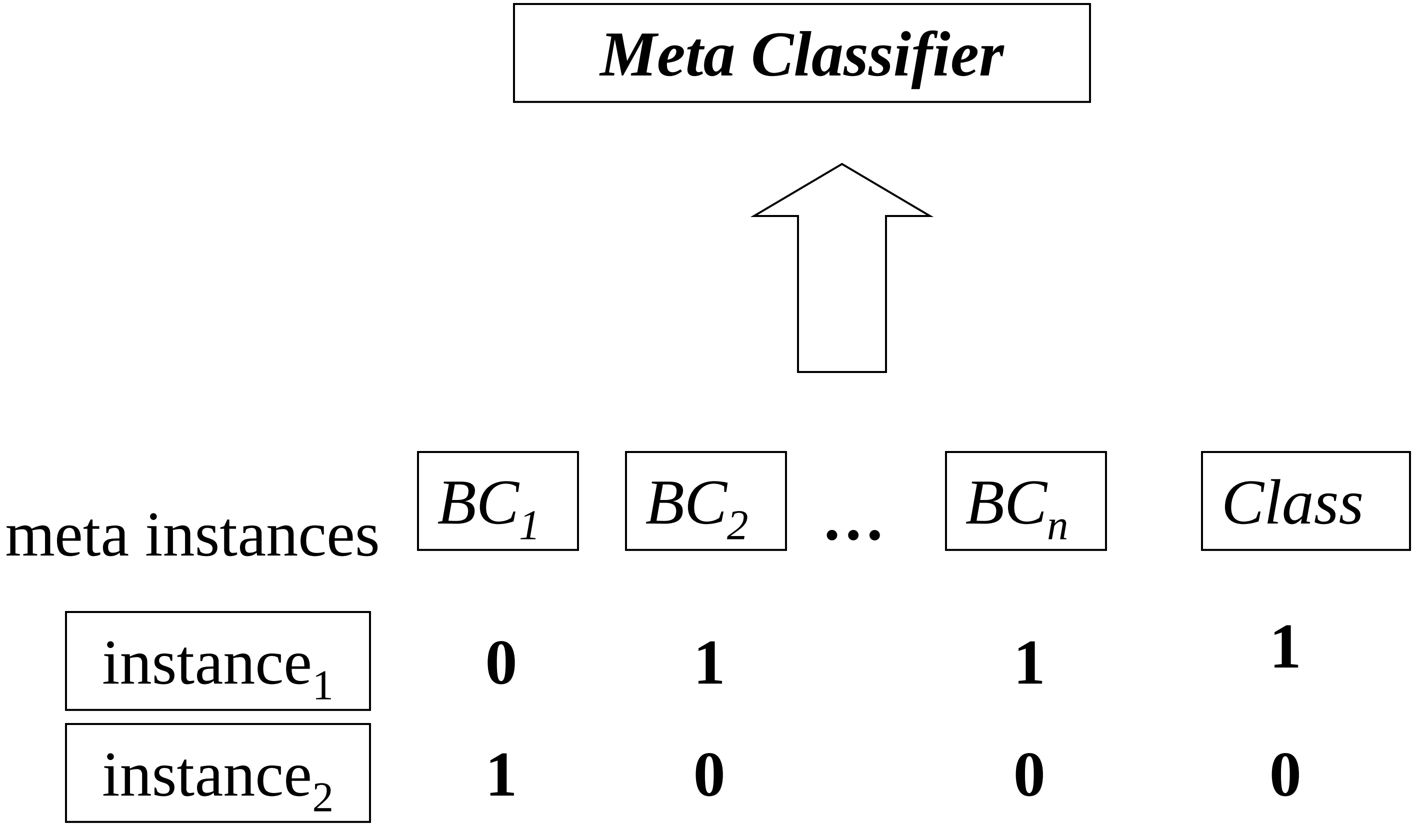
meta instances	BC_1	BC_2	…	BC_n	*Class*
instance_1	**0**	**1**		**1**	**1**

Stacking



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

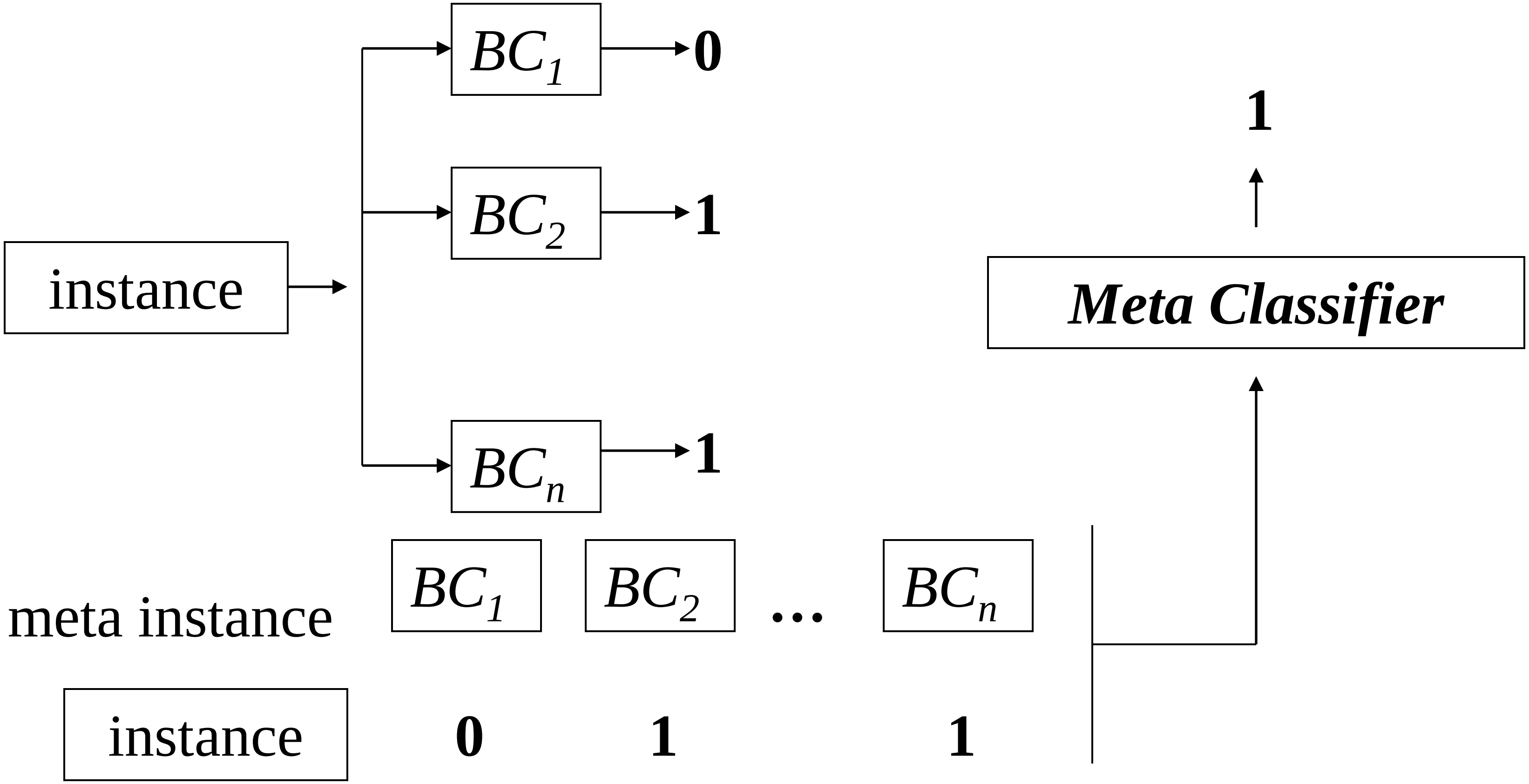
Stacking



Meta Classifier

meta instances	BC_1	BC_2	…	BC_n	*Class*
instance_1	**0**	**1**		**1**	**1**
instance_2	**1**	**0**		**0**	**0**

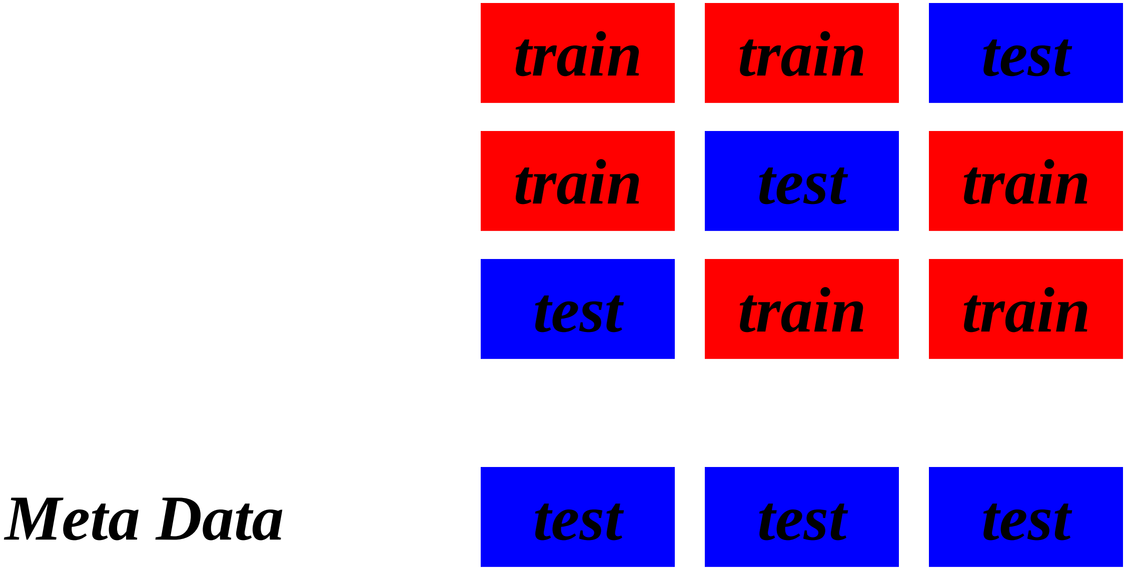
Stacking



BC1
BC2
BCn
0
1
1
instance
1
Meta Classifier
meta instance
BC1
BC2
…
BCn
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 fits training data will be chosen by the level-1 model! Thus,
- k-fold cross-validation-like scheme is employed! An example for k = 3!



	train	*train*	*test*
	train	*test*	*train*
	test	*train*	*train*
Meta Data	*test*	*test*	*test*

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!

Evolutionary Algorithms for Classifier Ensemble

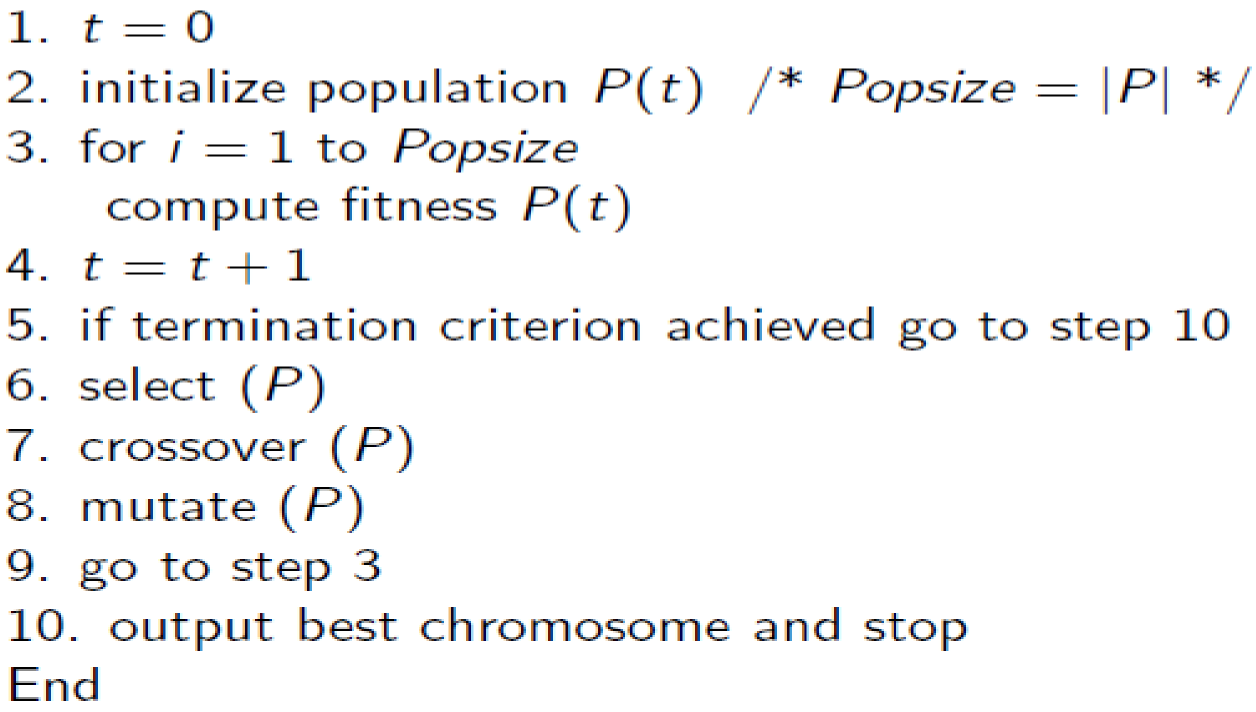
Genetic Algorithm: Quick Overview

- Randomized search and optimization technique
- Evolution produces good individuals, similar principles might work for solving complex problems
- Developed: USA in the 1970's by J. Holland
- Got popular in the late 1980's
- Early names: J. Holland, K. DeJong, D. Goldberg
- Based on ideas from *Darwinian Evolution*
- Can be used to solve a variety of problems that are not easy to solve using other techniques

Genetic Algorithm: Similarity with Nature

Genetic Algorithms	←→	Nature
A solution (phenotype)		Individual
Representation of a solution (*genotype*)		Chromosome
Components of the solution		Genes
Set of solutions		Population
Survival of the fittest (*Selection*)		Darwins theory
Search operators		Crossover and mutation
Iterative procedure		Generations

Basic Steps of Genetic Algorithm



```
1. $t = 0$
2. initialize population $P(t)$  /* $Popsize = |P|$ */
3. for $i = 1$ to $Popsize$
      compute fitness $P(t)$
4. $t = t + 1$
5. if termination criterion achieved go to step 10
6. select $(P)$
7. crossover $(P)$
8. mutate $(P)$
9. go to step 3
10. output best chromosome and stop
End
```

A. Ekbal and S. Saha (2011). Weighted Vote-Based Classifier Ensemble for Named Entity Recognition: A Genetic Algorithm-Based Approach. ACM Transactions on Asian Language Information Processing (ACM TALIP),Vol. 2(9),

DOI=10.1145/1967293.1967296

http://doi.acm.org/10.1145/1967293.1967296

Weighted Vote based Classifier Ensemble

- Motivation
  - All classifiers are not equally good at detecting all the classes

- Weighted voting: weights of voting vary among the classes for each classifier
  - *High*: Classes for which the classifier perform good
  - *Low*: Classes for which it's output is not very reliable
- *Crucial issue*: Selection of appropriate weights of votes per classifier

Problem Formulation

Let *no. of classifiers*=N, and *no. of classes*=M

Find the weights of votes V per classifier optimizing a function F(V)

-V: an real array of size N × M

-V(i , j) : weight of vote of the *i*th classifier for the *j*th class

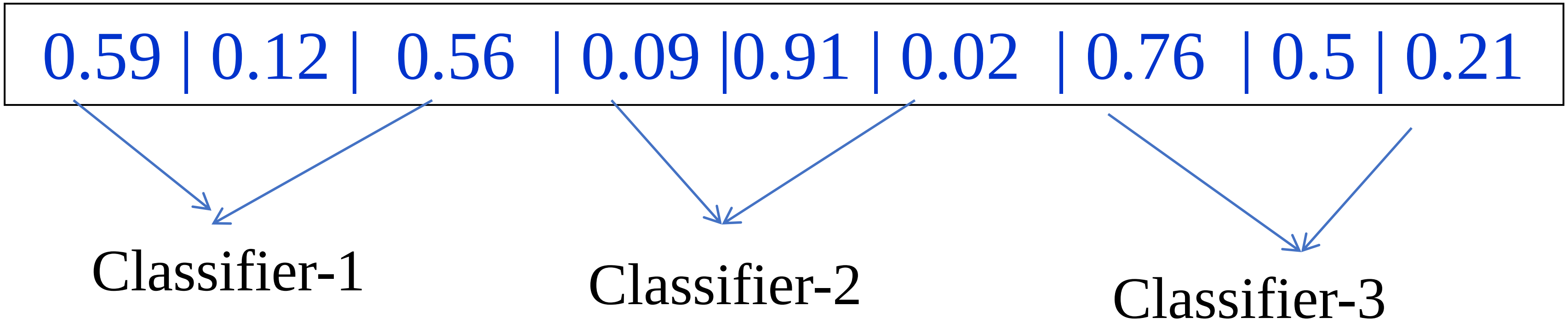
-V(i , j) ε [0, 1] denotes the degree of confidence of the *i*th classifier for the *j*th class

maximize F(B) ;

F ε {*recall, precision, F-measure*} and B is a subset of A

Here, *F1= F-measure*

Chromosome representation



0.59 | 0.12 | 0.56 | 0.09 |0.91 | 0.02 | 0.76 | 0.5 | 0.21

Classifier-1

Classifier-2

Classifier-3

- Real encoding used
- Entries of chromosome randomly initialized to a real (r) between 0 and 1: r = rand () / RAND_MAX+1
- If the population size P then all the P number of chromosomes of this population are initialized in the above way

Fitness Computation

Step-1: For M classifiers, F_i $i = 1$ to M be the F-measure values

Step-2: Train each classifier with 2/3 training data and evaluate with the remaining 1/3 part

Step-3: For ensemble output of the 1/3 test data, apply weighted voting on the outputs of M classifiers

(a). Weight of the output label provided by the *mth* classifier = I (m, i)

Here, *I(m, i) is the entry of the chromosome corresponding to mth* classifier and *ith class*

(b). Combined score of a class for a word *w*

$$f(c_i) = \sum I(m,i) \times F_m, \quad \forall m = 1 \text{ to } M \text{ and } op(w,m) = c_i$$

Fitness Computation

Op(w, m): output class produced by the *mth* classifier for word *w*

Class receiving the maximum score selected as joint decision

Step-4: Compute overall F-measure value for 1/3 data

Step-5: Steps 3 and 4 repeated to perform 3-fold cross validation

Step-6: Objective function or fitness function = F-measure$_{avg}$

Objective: Maximize the objective function using search capability of GA

Other Parameters

- Selection
  - Roulette wheel selection (*Holland, 1975; Goldberg, 1989*)
- Crossover
  - Normal Single-point crossover (Holland, 1975)
- Mutation
  - Probability selected adaptively (*Srinivas and Patnaik, 1994*)
  - Helps GA to come out from local optimum

Termination Condition

- Execute the processes of *fitness computation*, *selection*, *crossover*, and *mutation* for a maximum number of generations
- *Best solution*-Best string seen up to the last generation

- Best solution indicates
  - Optimal voting weights for all classes in each classifier

- Elitism implemented at each generation
  - Preserve the best string seen up to that generation in a location outside the population
  - Contains the most suitable classifier ensemble

What is Named Entity Recognition and Classification (NERC)?

- NERC – Named Entity Recognition and Classification (NERC) involves identification of proper names in texts, and classification into a set of pre-defined categories of interest as:
  - Person names (names of people)
  - Organization names (companies, government organizations, committees, etc.)
  - Location names (cities, countries etc)
  - Miscellaneous names (Date, time, number, percentage, monetary expressions, number expressions and measurement expressions)

Named Entity Recognition

Markables (as defined in MUC6 and MUC7)

- Names of **organization**, **person**, **location**
- Mentions of **date** and **time**, **money** and **percentage**

Example:

"Ms. **Washington**'s candidacy is being championed by several powerful lawmakers including her boss, Chairman **John Dingell** (D., **Mich**.) of the **House Energy and Commerce Committee**."

NE Features: Mostly language independent

NE Features

- Context Word: Preceding and succeeding words
- Word Suffix
  - Not necessarily linguistic suffixes
  - Fixed length character strings stripped from the endings of words
  - Variable length suffix -binary valued feature
- Word Prefix
  - Fixed length character strings stripped from the beginning of the words
- Named Entity Information: Dynamic NE tag (s) of the previous word (s)

NE Features

- First Word (binary valued feature): Check whether the current token is the first word in the sentence
- Length (binary valued): Check whether the length of the current word less than three or not (shorter words rarely NEs)
- Position (binary valued): Position of the word in the sentence
- Infrequent (binary valued): Infrequent words in the training corpus most probably NEs

NE Features

- Digit features: Binary-valued
  - Presence and/or the exact number of digits in a token
    - CntDgt : Token contains digits
    - FourDgt: Token consists of four digits
    - TwoDgt: Token consists of two digits
    - CnsDgt: Token consists of digits only

- Combination of digits and punctuation symbols
  - CntDgtCma: Token consists of digits and comma
  - CntDgtPrd: Token consists of digits and periods

NE Features

- Combination of digits and symbols
  - CntDgtSlsh: Token consists of digit and slash
  - CntDgtHph: Token consists of digits and hyphen
  - CntDgtPrctg: Token consists of digits and percentages

- Combination of digit and special symbols
  - CntDgtSpl: Token consists of digit and special symbol such as $, # etc.

NE Features

- Part of Speech (POS) Information: POS tag(s) of the current and/or the surrounding word(s)
  - SVM-based POS tagger (Ekbal and Bandyopadhyay, 2008)
  - SVM based NERC→POS tagger developed with a fine-grained tagset of 27 tags
  - Coarse-grained POS tagger
    - Nominal, PREP (Postpositions) and Other

- Gazetteer based features (binary valued): Several features extracted from the gazetteers

Datasets

- Web-based Bengali news Corpus (Ekbal and Bandyopadhyay, 2008, *Language Resources and Evaluation of Springer*)
  - *34 million* wordforms
  - News data collection of 5 years
- NE annotated corpus for Bengali
  - Manually annotated 250K wordforms
  - IJCNLP-08 Shared Task on NER for South and South East Asian Languages (available at http://ltrc.iiit.ac.in/ner-ssea-08)
- NE annotated datasets for Hindi and Telugu
  - NERSSEAL shared task

NE Tagset

- Reference Point- CoNLL 2003 shared task tagset
- Tagset: 4 NE tags
  - Person name
  - Location name
  - Organization name
  - Miscellaneous name (*date*, *time*, *number*, *percentages*, *monetary expressions* and *measurement expressions*)

- IJCNLP-08 NERSSEAL Shared Task Tagset: Fine-grained 12 NE tags (available at http://ltrc.iiit.ac.in/ner-ssea-08)

- Tagset Mapping (12 NE tags→4 NE tags)
  - ❑NEP → Person name
  - ❑NEL→ Location name
  - ❑NEO→ Organization name
  - ❑NEN [number], NEM [Measurement] and NETI [time]→Miscellaneous name
  - ❑NETO [title-object], NETE [term expression], NED [designations], NEA [abbreviations], NEB [brand names], NETP [title persons

Training and Test Datasets

Language	#Words in training	#NEs in training	#Words in test	#NEs in test
Bengali	312,947	37,009	37,053	4,413
Hindi	444,231	26,432	32,796	58,682
Telugu	57,179	4,470	6,847	662
Oriya	93,573	4,477	2,183	206

Experiments

- Classifiers used
  - Maximum Entropy (ME): Java based OpenNLP package (http://maxent.sourceforge.net/)
  - Conditional Random Field: C++ based CRF++ package (http://crfpp.sourceforge.net/)
  - Support Vector Machine:
    - YamCha toolkit

      (http://chasen-org/ taku/software/yamcha/)
    - TinySVM-0.07

      (http://cl.aist-nara.ac.jp/ taku-ku/software/TinySVM)
    - Polynomial kernel function

Experiments

- GA: population size=50, number of generations=40, mutation and crossover probabilities are selected adaptively.

- Baselines
  - Baseline 1: Majority voting of all classifiers
  - Baseline 2: Weighted voting of all classifiers (*weight*: overall average F-measure value)
  - Baseline 3: Weighted voting of all classifiers (*weight*: F-measure value of the individual class)

Results (*Bengali*)

Model	Recall	Precision	F-measure
Best Individual Classifier	89.42	90.55	89.98
Baseline-1	84.83	85.90	85.36
Baseline-2	85.25	86.97	86.97
Baseline-3	86.97	87.34	87.15
Stacking	90.17	91.74	90.95
ECOC	89.78	90.89	90.33
QBC	90.01	91.09	90.55
GA based ensemble	92.08	92.22	92.15

Results (*Hindi*)

Model	**Recall**	**Precision**	**F-measure**
Best Individual Classifier	88.72	90.10	89.40
Baseline-1	63.32	90.99	74.69
Baseline-2	74.67	94.73	83.64
Baseline-3	75.52	96.13	84.59
Stacking	89.80	90.61	90.20
ECOC	90.16	91.11	90.63
GA based ensemble	96.07	88.63	92.20

Results (*Telugu*)

Model	Recall	Precision	F-measure
Best Individual Classifier	77.42	77.99	77.70
Baseline-1	60.12	87.39	71.23
Baseline-2	71.87	92.33	80.33
Baseline-3	72.22	93.10	81.34
Stacking	77.65	84.12	80.76
ECOC	77.96	85.12	81.38
GA based ensemble	78.82	91.26	84.59

Results (*Oriya*)

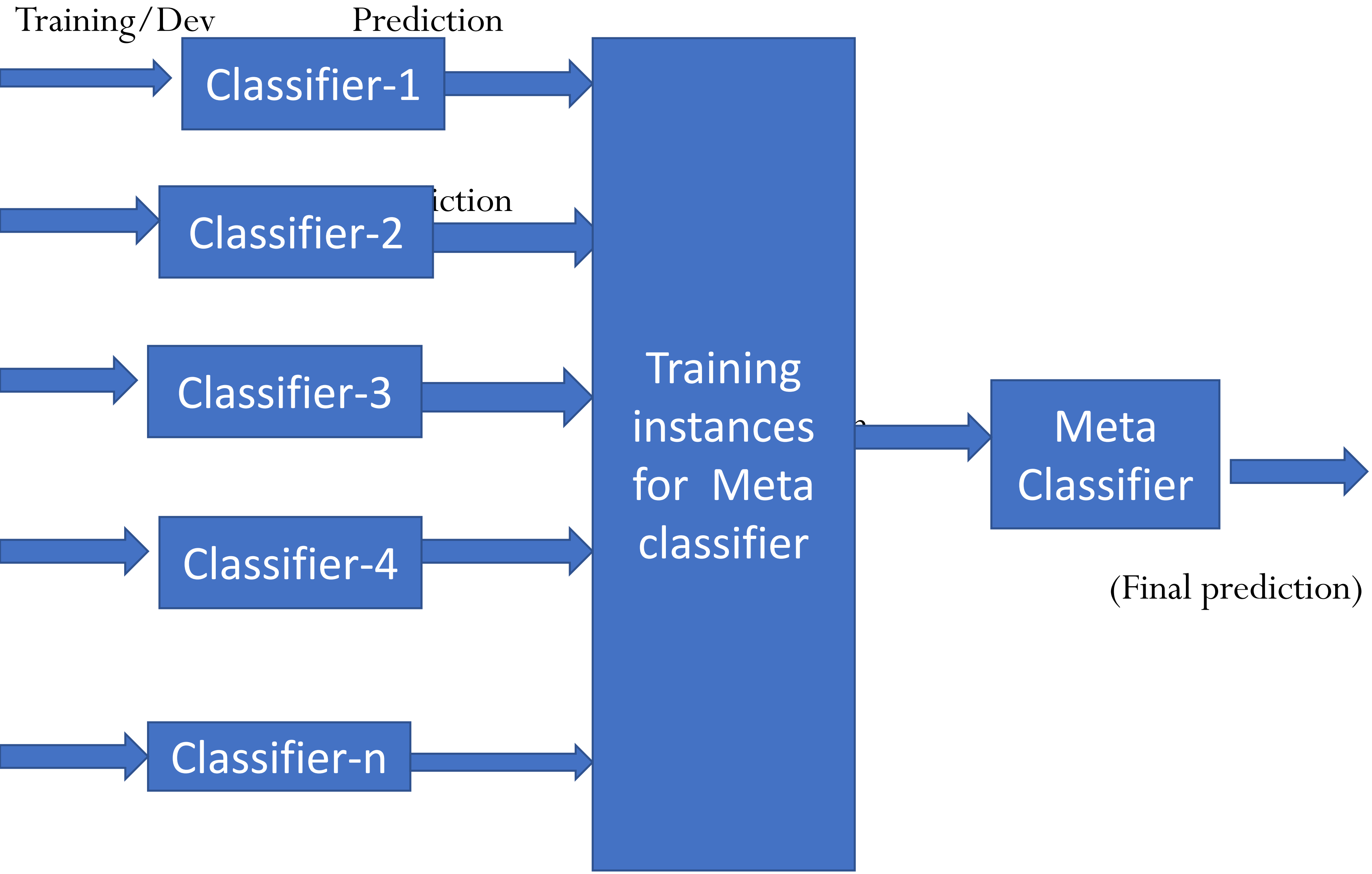
Model	Recall	Precision	F-measure
Best Individual Classifier	86.55	88.03	87.29
Baseline-1	86.95	88.33	87.63
Baseline-2	87.12	88.50	87.80
Baseline-3	87.62	89.12	88.36
Stacking	87.90	89.53	88.71
ECOC	87.04	88.56	87.79
GA based ensemble	88.56	89.98	89.26

Results (*English*)

Model	Recall	Precision	F-measure
Best Individual Classifier	86.16	85.24	86.31
Baseline-1	85.75	86.12	85.93
Baseline-2	86.20	87.02	86.61
Baseline-3	86.65	87.25	86.95
Stacking	85.93	86.45	86.18
ECOC	86.12	85.34	85.72
GA based ensemble	88.72	88.64	88.68

Asif Ekbal and Sriparna Saha (2013). Stacked ensemble coupled with feature selection for biomedical entity extraction, ***Knowledge Based Systems****, volume (46), PP. 22–32, Elsevier.*

Stacked Model with Feature Selection



Training/Dev

Prediction

Classifier-1

iction

Classifier-2

Classifier-3

Classifier-4

Classifier-n

Training instances for Meta classifier

Meta Classifier

(Final prediction)

Stacked Model with Feature Selection

- Feature selection
  - GA based
  - Select few classifiers from the final population
  - Term them as base classifiers (CRF and SVM)
- Train the base classifiers
- Evaluate on the development data
- Meta-level training instances
  - Predictions obtained on the development data
  - Original attributes

Stacked Model with Feature Selection

- For the test set
  - Generate predictions from the base classifiers
  - Use these predictions along with the original attributes as features
- Meta classifier- CRF

Experiments (JNLPBA-2004)

Model	Recall	Precision	F-measure
Best individual classifier	73.10	76.78	74.90
Majority ensemble	71.03	75.76	73.32
Weighted ensemble	71.42	75.90	73.59
Stacked ensemble	75.15	75.20	75.17

At par the state-of-the-art system

Experiments (GENETAG)

Model	Recall	Precision	F-measure
Best individual classifier	94.41	93.50	93.95
Majority ensemble	94.45	93.65	94.05
Weighted ensemble	94.67	93.91	94.29
Stacked ensemble	95.12	94.29	94.70

At par the state-of-the-art system