Imbalanced Learning

Classification facing the real world

Hao Ding

dinghao@stu.ouc.edu.cn

Table of contents

1. Classification

2. Imbalanced Learning

3. Solutions

4. Methods

5. Evaluation Criteria

Classification

Process

Original data

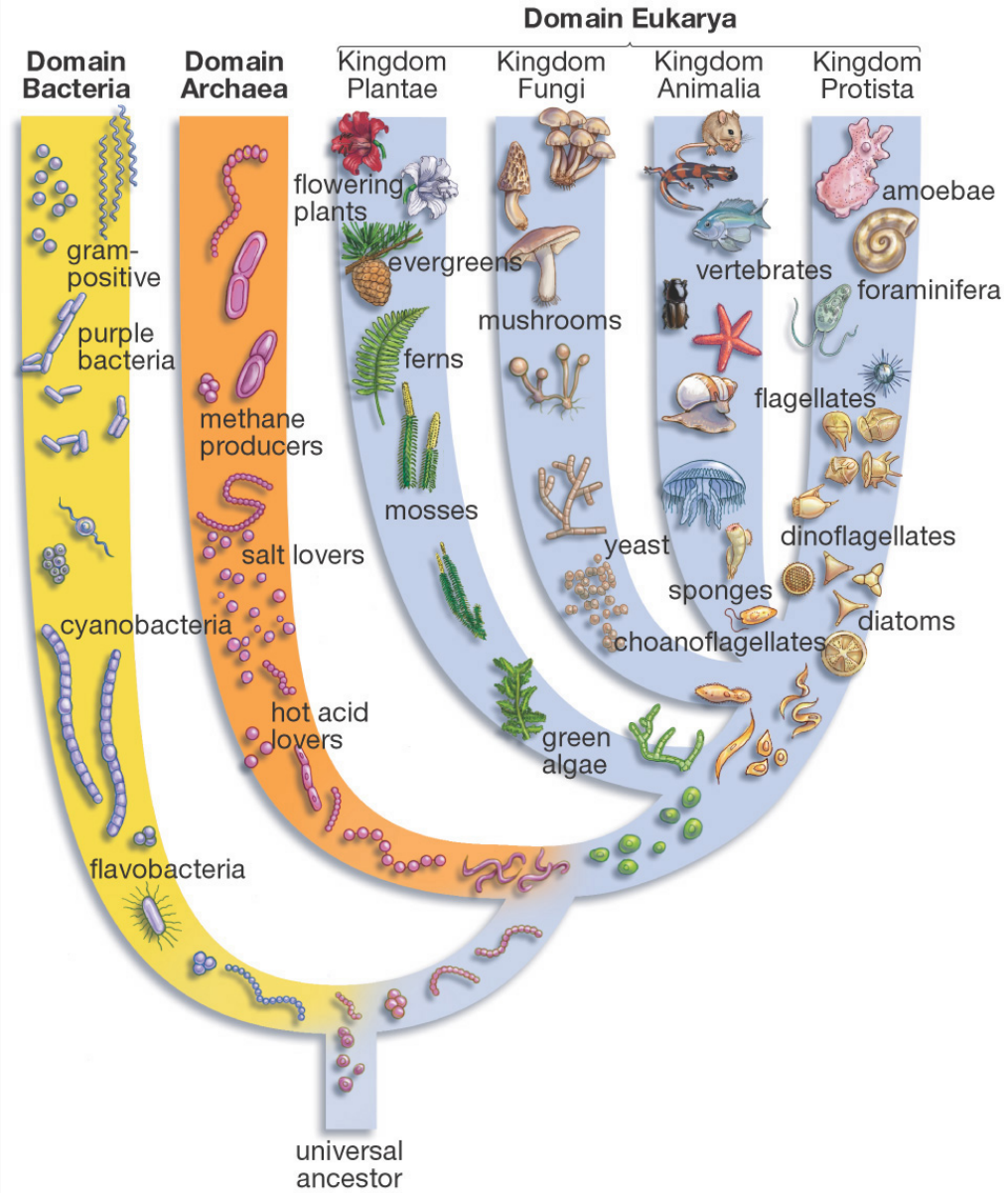
Preprocessing

Classification

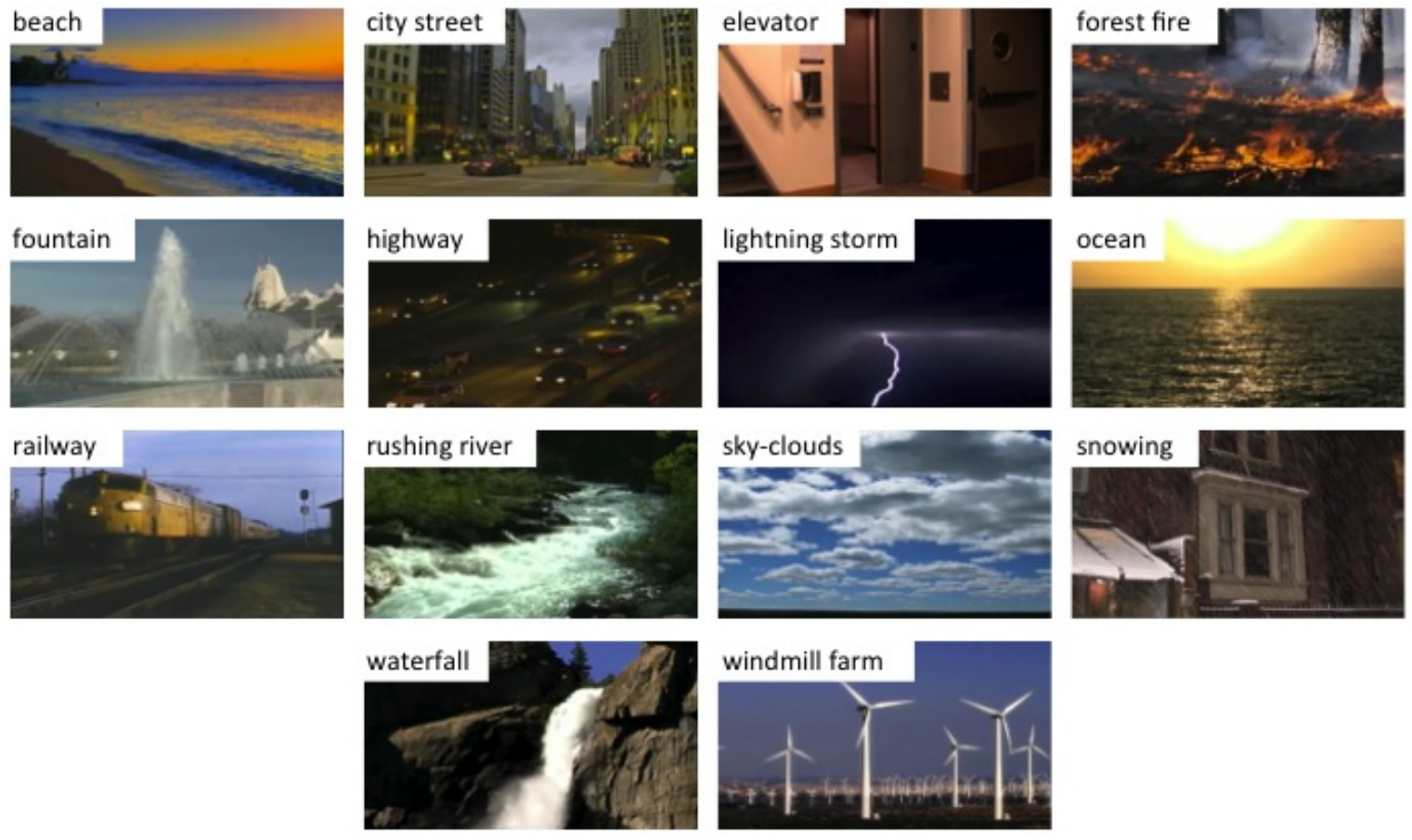
Preprocess

- image segmentation
- image enhancement and restoration
- image dehazing and deblurring
- image depth estimation
- some feature process

Classification of Living Things



Scene classification



K.G. Derpanis, M. Lecce, K. Daniildis and R.P. Wildes, Dynamic Scene Understanding: The Role of Orientation Features in Space and Time in Scene Classification, IEEE (CVPR '12)

Face recognition

Ruogu Fang, Kevin D. Tang, Noah Snavely, Tsuhan Chen. Towards Computational Models of Kinship Verification. IEEE (ICIP '10)

Problems

✓ Explosive availability of raw data
✓ Well-developed algorithms for data analysis

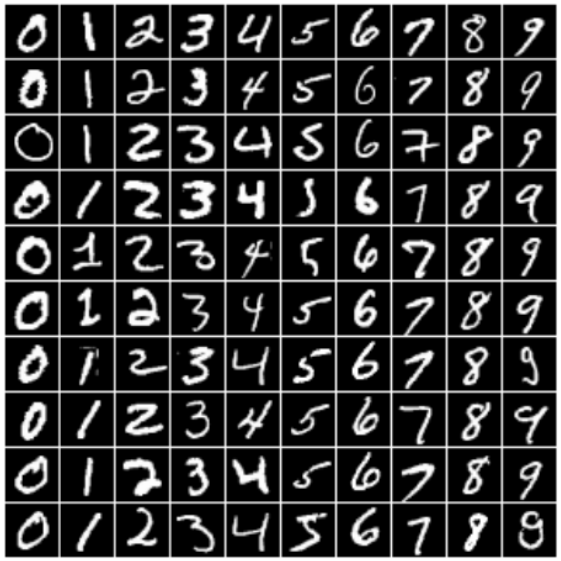
Requirement?

- Balanced distribution of data
- Equal costs of misclassification

Datasets

MNIST(1998)：

Number of training images：60000
Number of testing images：10000
Number of categories: 10



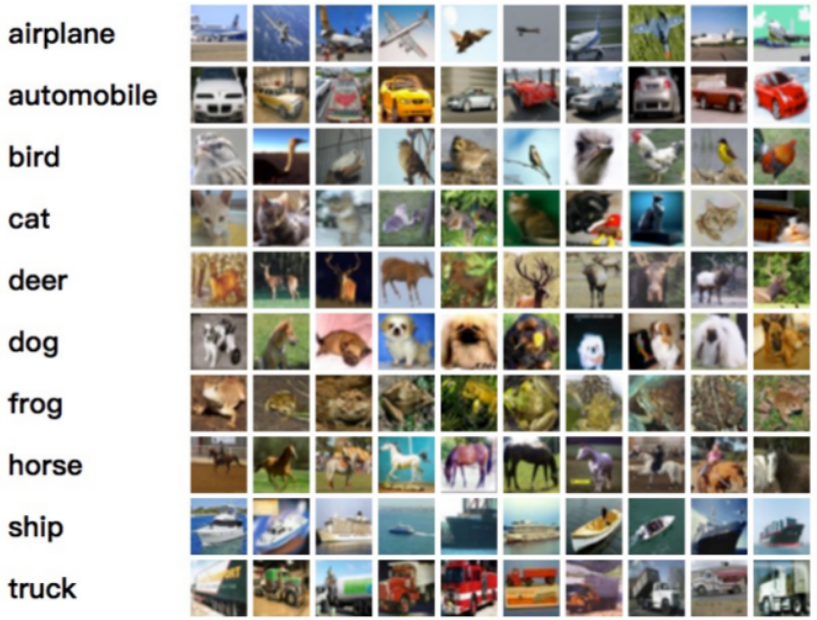
http://yann.lecun.com/exdb/mnist/

Datasets

CIFAR-10:

Number of training images: 50000
Number of testing images: 10000
Number of categories: 10

CIFAR-100:
Number of training images: 50000
Number of testing images: 10000
Number of categories: 100



https://www.cs.toronto.edu/ kriz/cifar.html

Datasets

Caltech-101:

Number of images：9144
Number of categories: 102

Caltech-256:

Number of images：30607
Number of categories: 257

http://www.vision.caltech.edu/Image_Datasets/Caltech256/

✓ Explosive availability of raw data
✓ Well-developed algorithms for data analysis

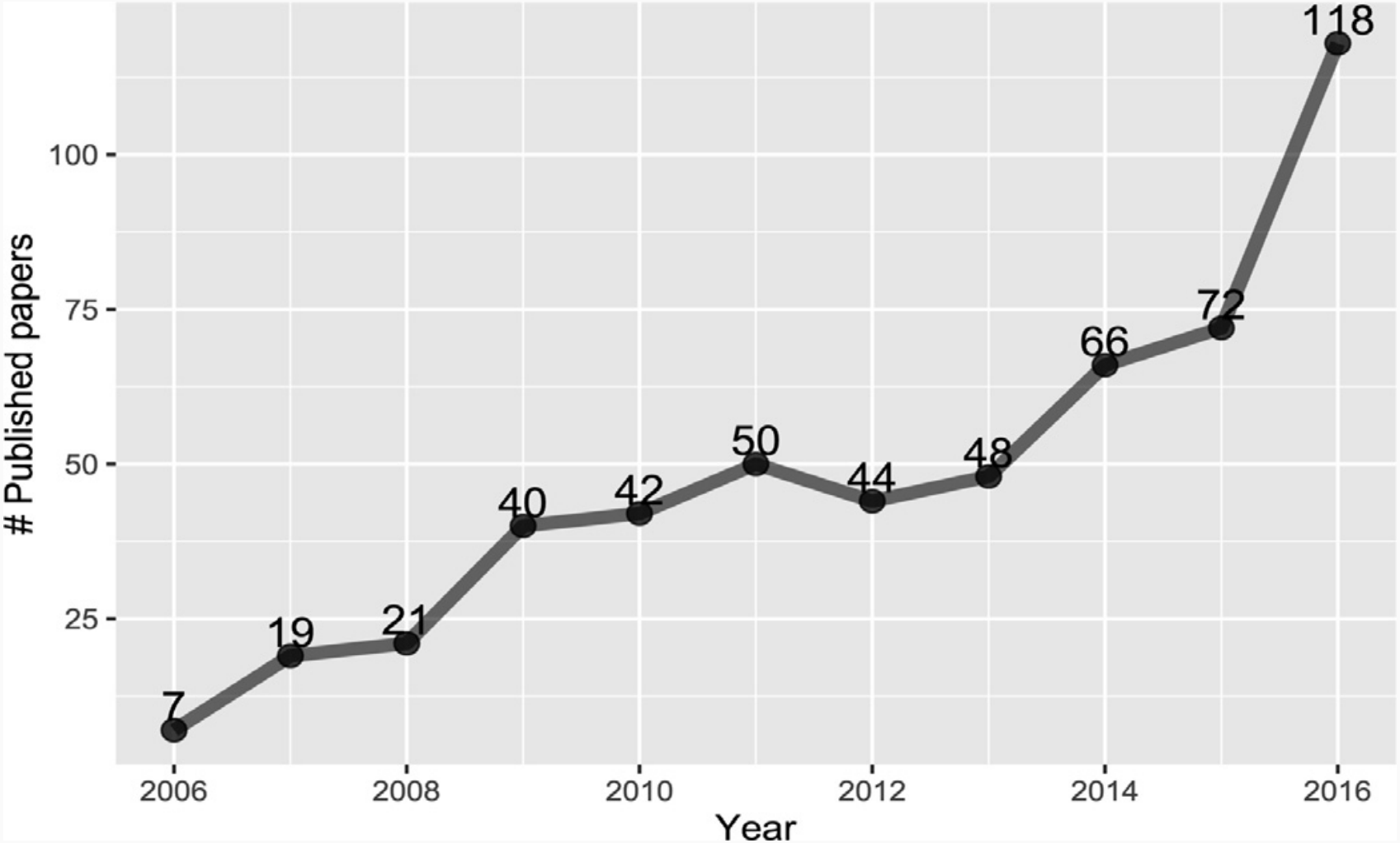
Requirement?

- Balanced distribution of data
- Equal costs of misclassification

What about data in reality?

Imbalanced Learning

Growing interest



Phishing problem

1. For 1 hour, google collects 1M e-mails randomly
2. Persume they have labled them as "phishing" or "not-phishing"
3. You get the data
4. Try out a few of your favorite classifiers
5. Congretulations and you achieved an accuracy of 99.997%

Phishing problem

1. For 1 hour, google collects 1M e-mails randomly
2. Persume they have labled them as "phishing" or "not-phishing"
3. You get the data
4. Try out a few of your favorite classifiers
5. Congretulations and you achieved an accuracy of 99.997%

Should you be happy?

Phishing problem

The phishing problem is what is called an imbalanced data problem

This occurs where there is a large discrepancy between the number of examples with each class label

e.g. for our 1M example dataset only about 30 would actually represent phishing e-mails

What is probably going on with our classifier?

Phishing problem

Answer:

Always predict not-phishing $\rightarrow$ 99.997% accuracy

Phishing problem

Why does the classifier learn this?

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} Cost(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

- Many classifiers are designed to optimize error/accuracy
- This tends to bias performance towards the majority class
- Anytime there is an imbalance in the data this can happen
- It is particularly pronounced, though, when the imbalance is more pronounced

Imbalance rate

Imbalance rate

Ratio of size of the majority class to minority class

imbalance rate in this case equals to:

$$999970/30 \approx 33332.33$$

In some cases, the value can be as huge as 10^6

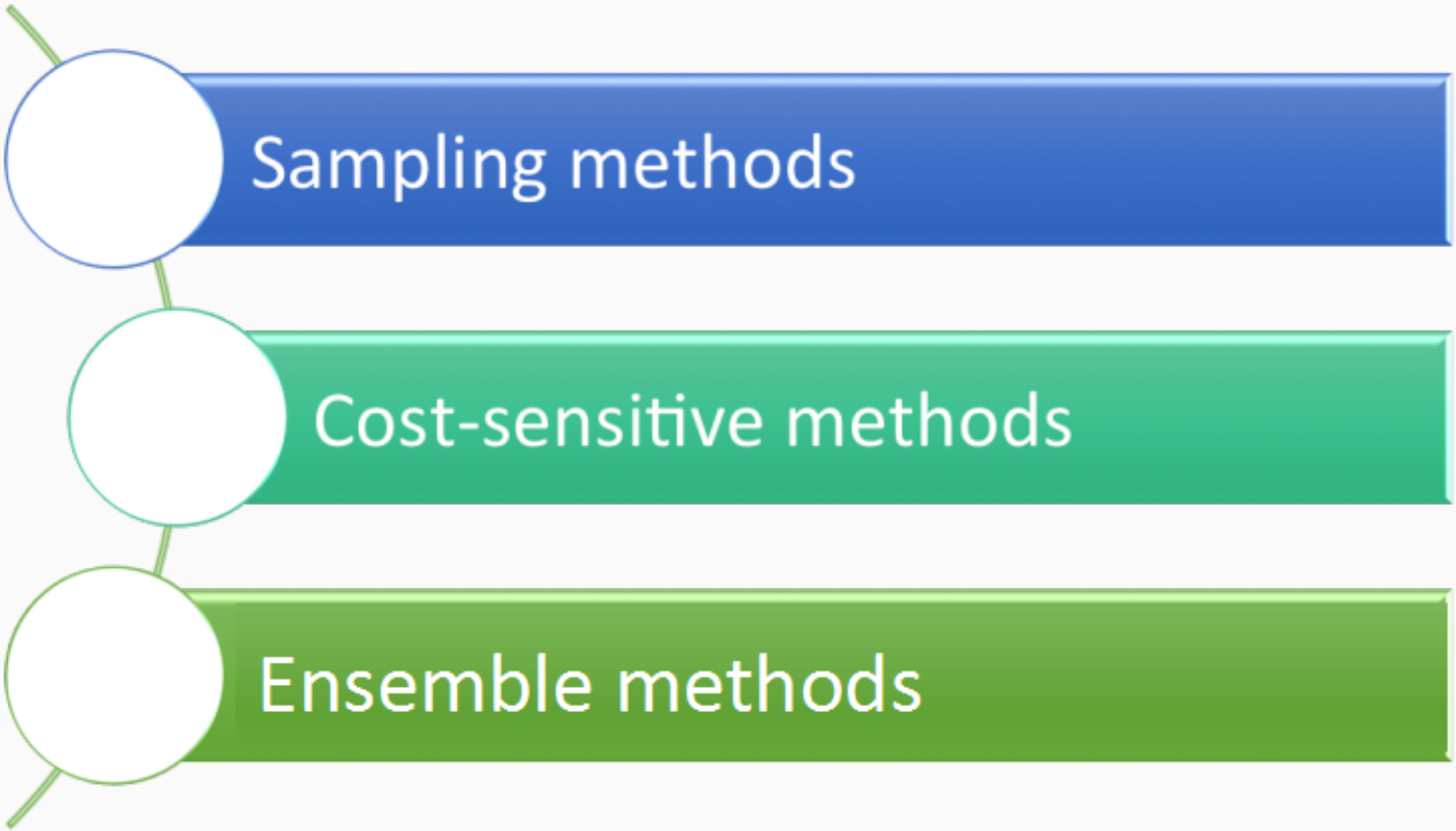
Imbalanced problem domains

- Medical diagnosis
- Predicting faults/failures (e.g. hard-drive failures, mechanical failures, etc.)
- Predicting rare events (e.g. earthquakes)
- Detecting fraud (credit card transactions, internet traffic)
- Plankton image classification
- Many other classification problems

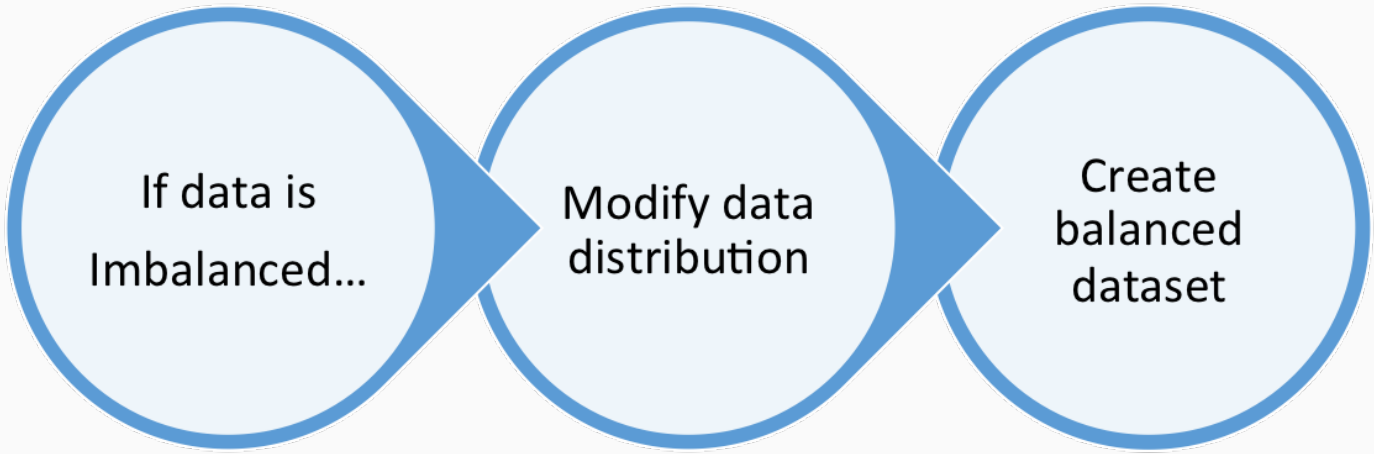
Imbalanced problem

Solutions

Solutions to imbalanced learning

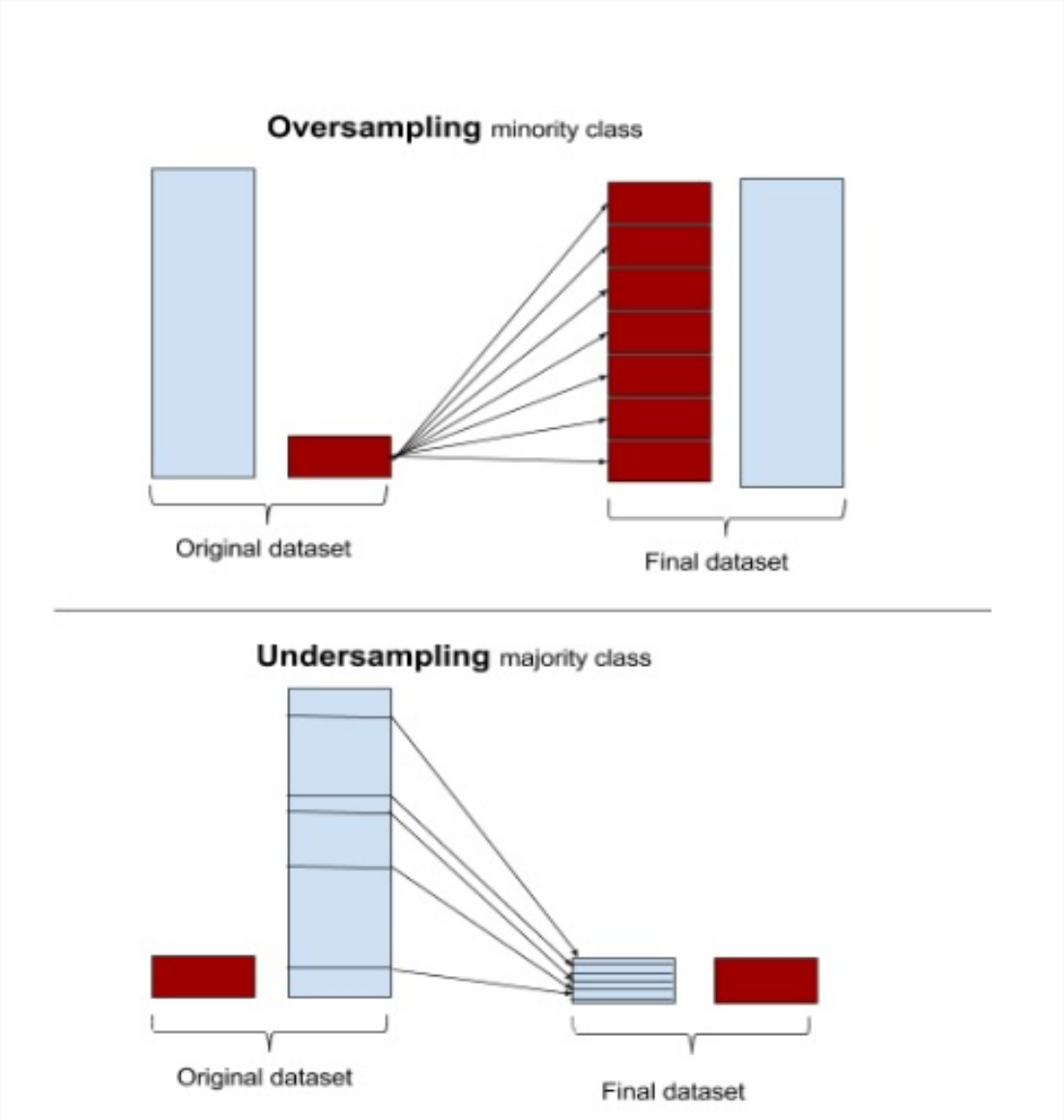


Sampling methods

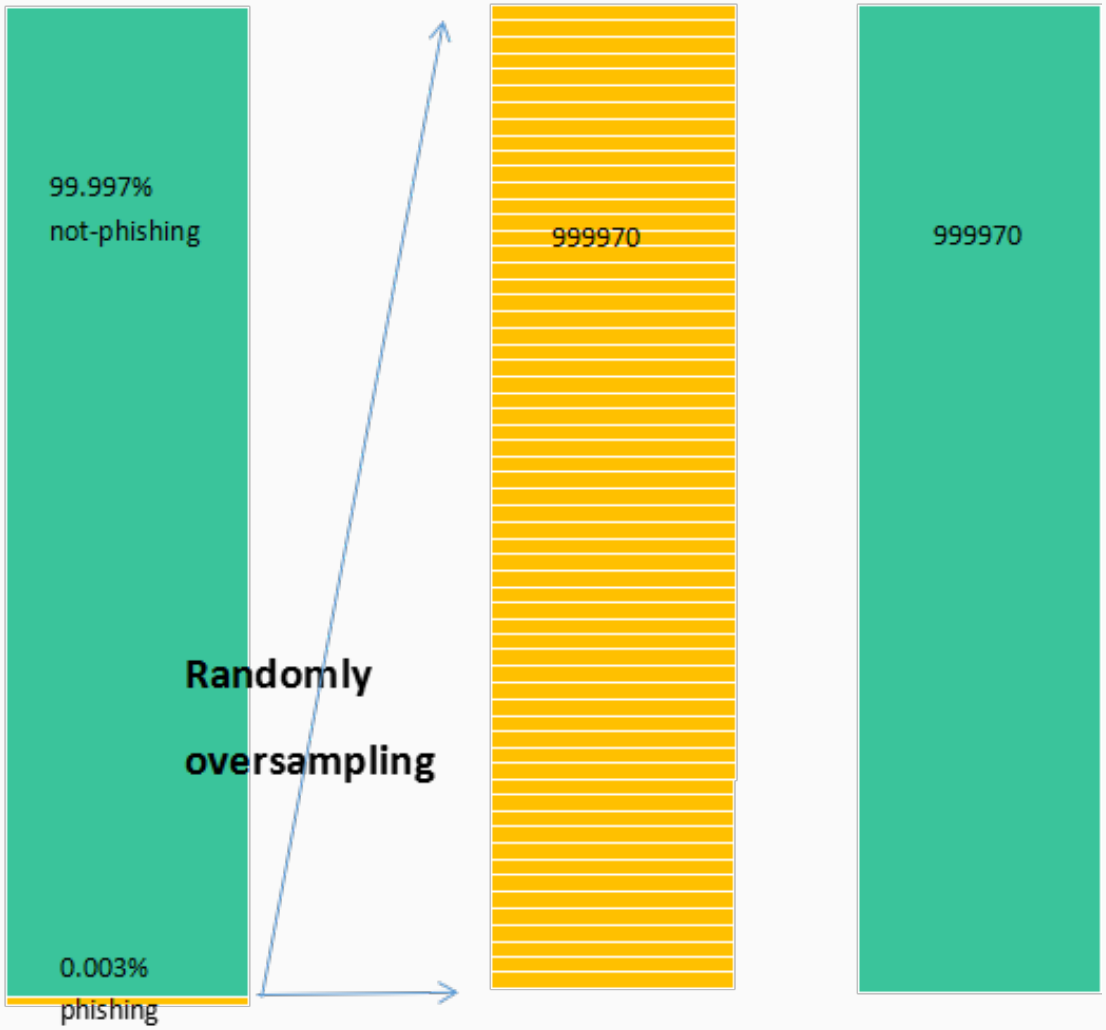


Source: H. He and E. A. Garcia, Learning from Imbalanced Data, IEEE Trans. Knowledge and Data Engineering, vol. 21, issue 9, pp. 1263-1284, 2009

Sampling methods



Over-Sampling



Over-Sampling

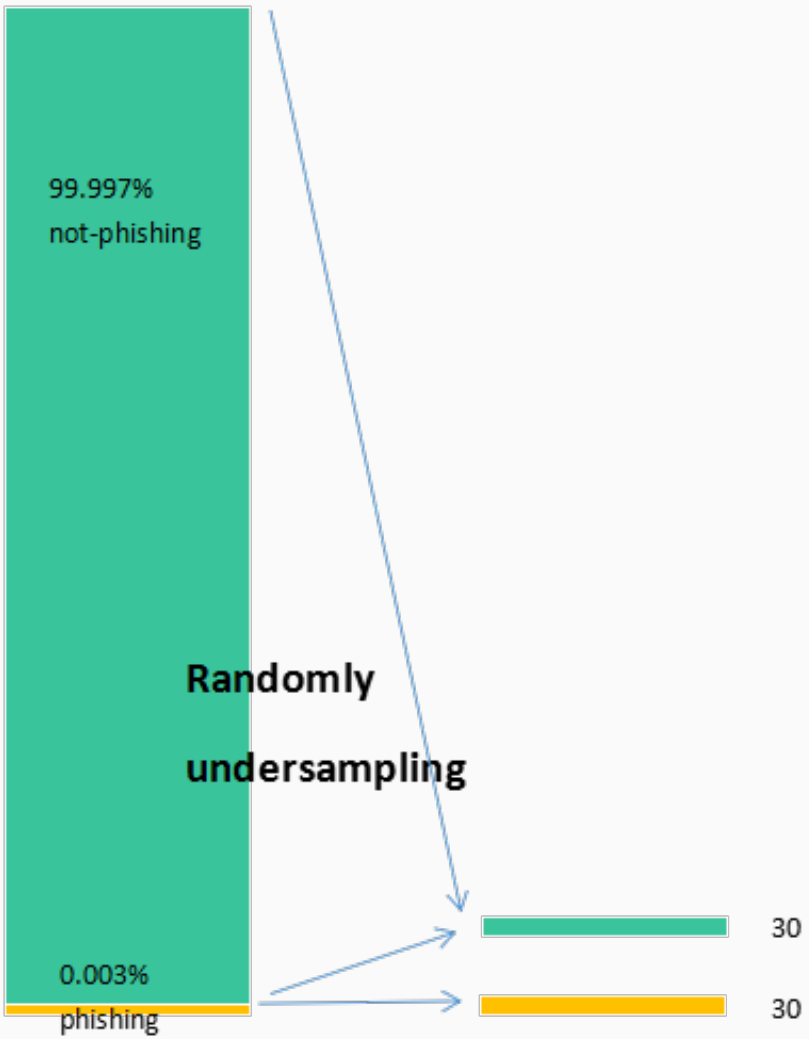
Pros:

- Easy to implement
- Utilizes all of the training data
- Tends to perform well in a broader set of circumstances than subsampling

Cons:

- Computationally expensive to train classifier

Under-Sampling



Under-Sampling

Pros:

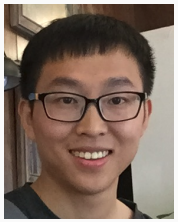
- Easy to implement
- Training becomes much more efficient(smaller training set)
- For some domains, can work very well

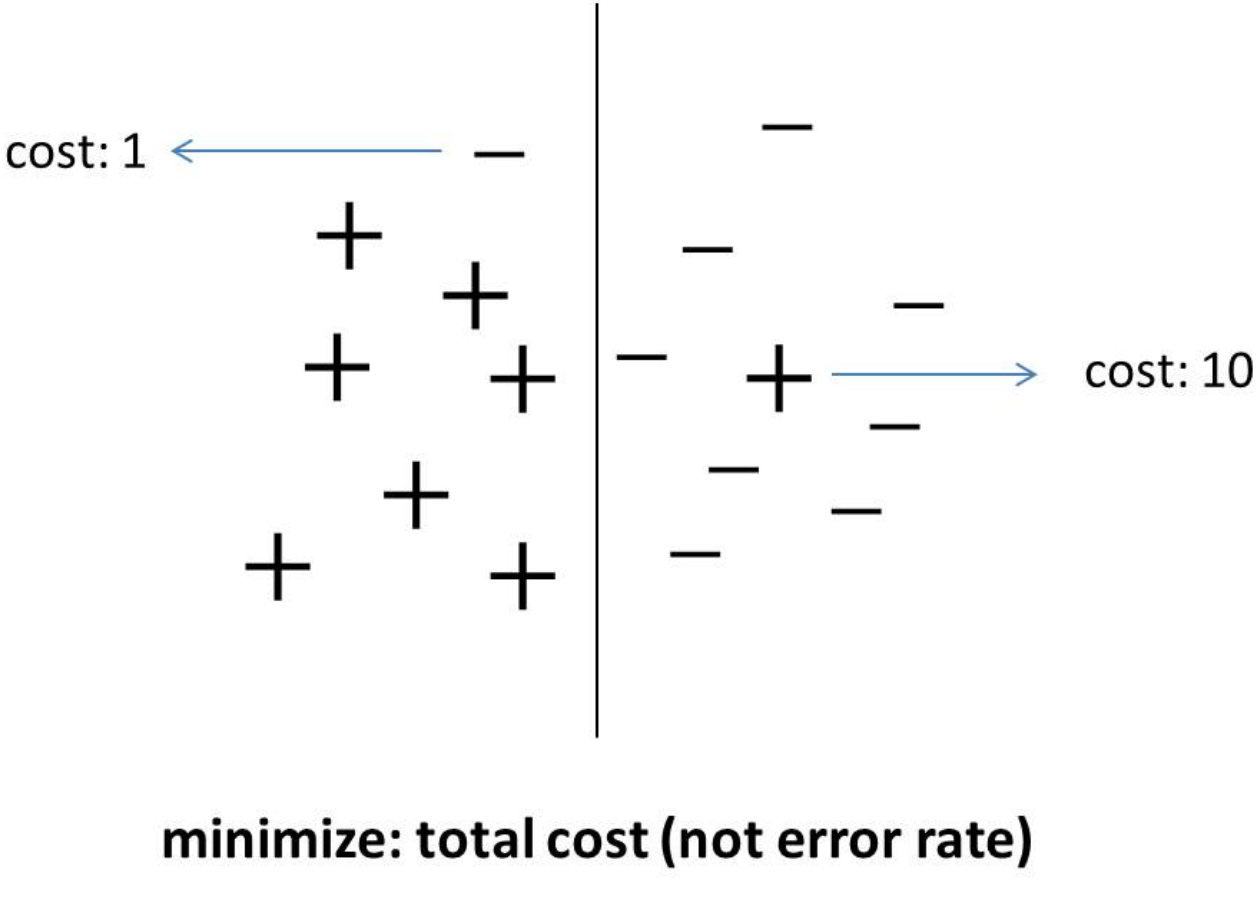
Cons:

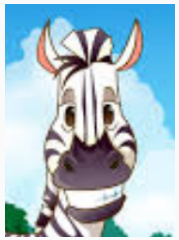
- Throwing away a lot of data/information

Cost-sensitive methods

Cost-sensitive methods







Cost-sensitive methods

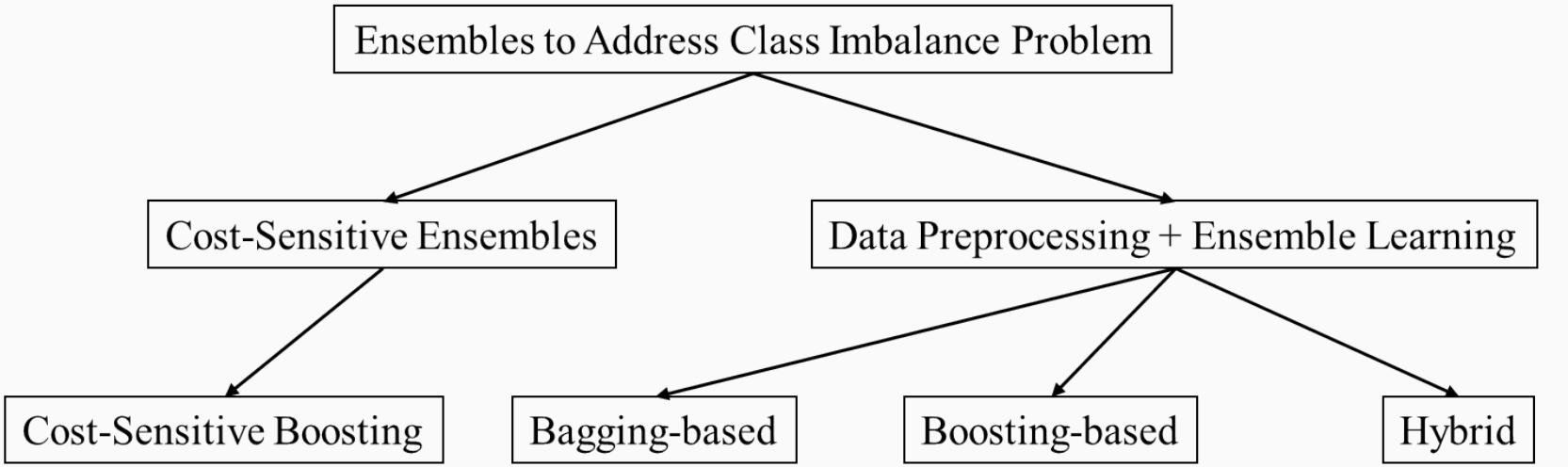
Pros:

- Achieves the effect of oversampling without the computational cost
- Utilizes all of the training data
- Tends to perform well in a broader set circumstances

Cons:

- Requires a classifier that can deal with weights

Ensemble methods



Methods

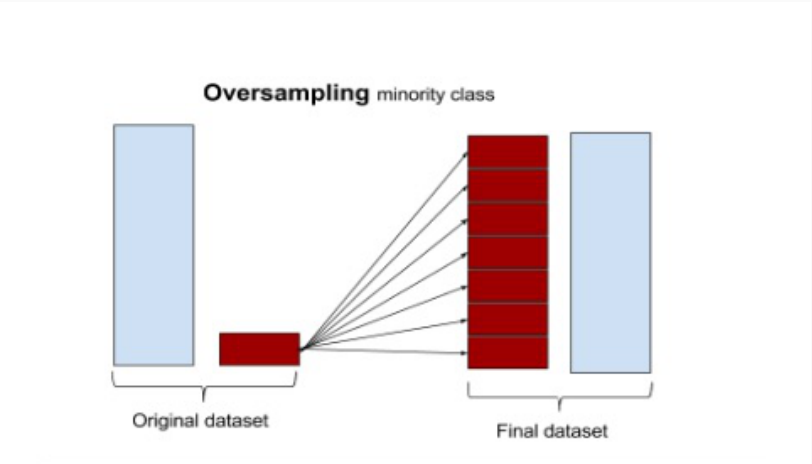
Random Sampling

S: training data set; S_{min}: set of minority class samples;

S_{maj}: set of majority class samples; E: generated samples

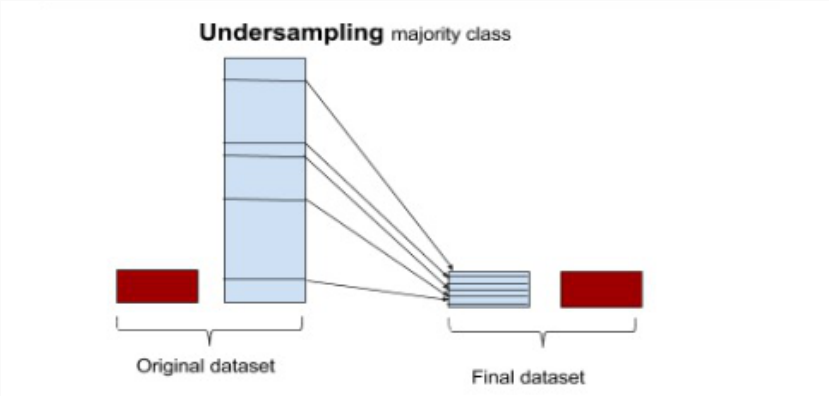
Random oversampling

- Expand the minority
- $|S'_{min}| \leftarrow |S_{min}| + |E|$
- $|S'| \leftarrow |S_{min}| + |S_{maj}| + |E|$
- Overfitting due to multiple "tied" instance



Random undersampling

- shrink the majority
- $|S'_{maj}| \leftarrow |S_{maj}| - |E|$
- $|S'| \leftarrow |S_{min}| + |S_{maj}| - |E|$
- Loss of important concepts



Informed Undersampling

EasyEnsemble

Unsupervised: use random subsets of the majority class to create balance and form multiple classifiers

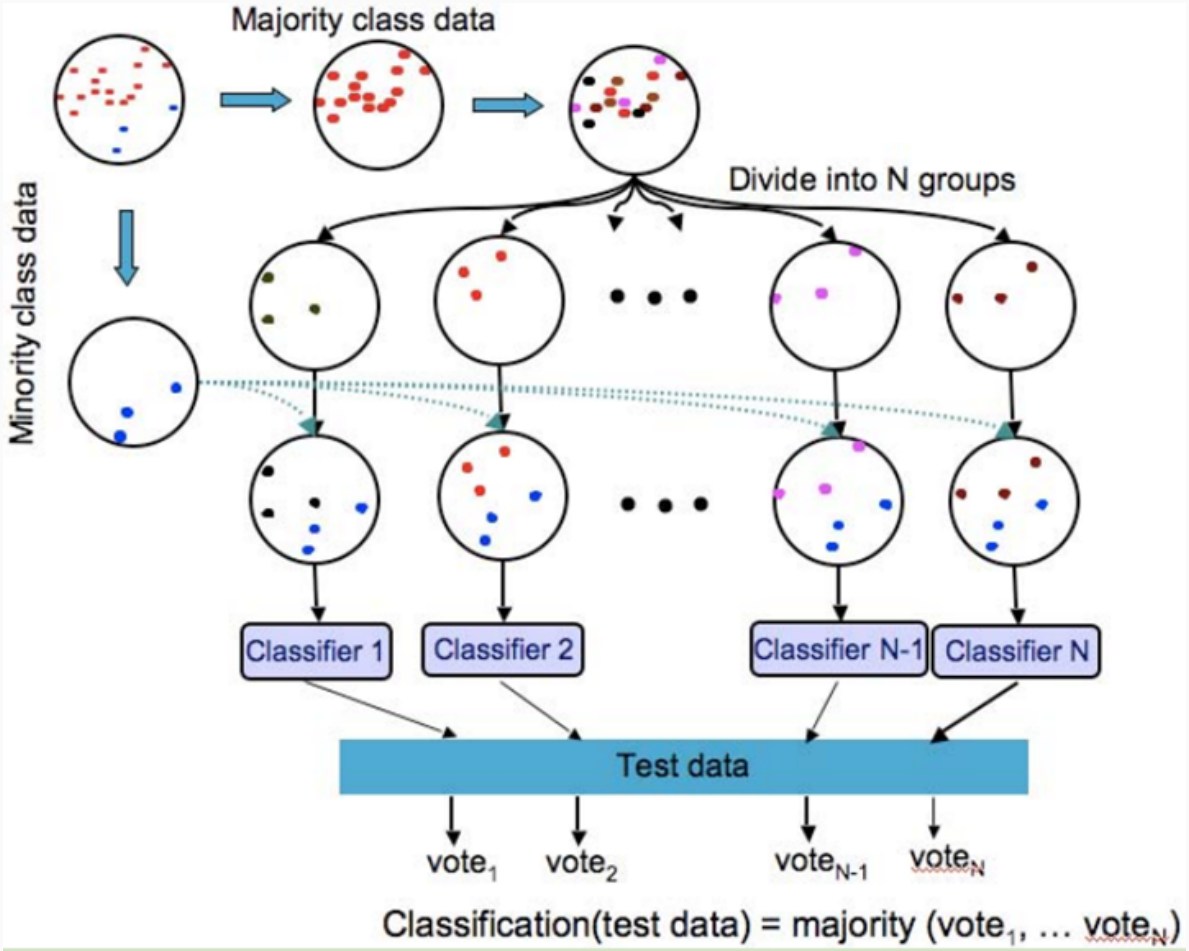
BalanceCascade

Supervised: iteratively create balance and pull out redundant samples in majority class to form a final classifier

X. Y. Liu, J. Wu, and Z. H. Zhou, Exploratory under-sampling for class-imbalance learning, in Proceedings of International Conference on Data Mining, pp. 965969, IEEE Computer Society, 2006.

Informed Undersampling

EasyEnsemble



X. Y. Liu, J. Wu, and Z. H. Zhou, Exploratory under-sampling for class-imbalance learning, in Proceedings of International Conference on Data Mining, pp. 965969, IEEE Computer Society, 2006.

Informed Undersampling

EasyEnsemble

Pros:

- Save majority samples

Cons:

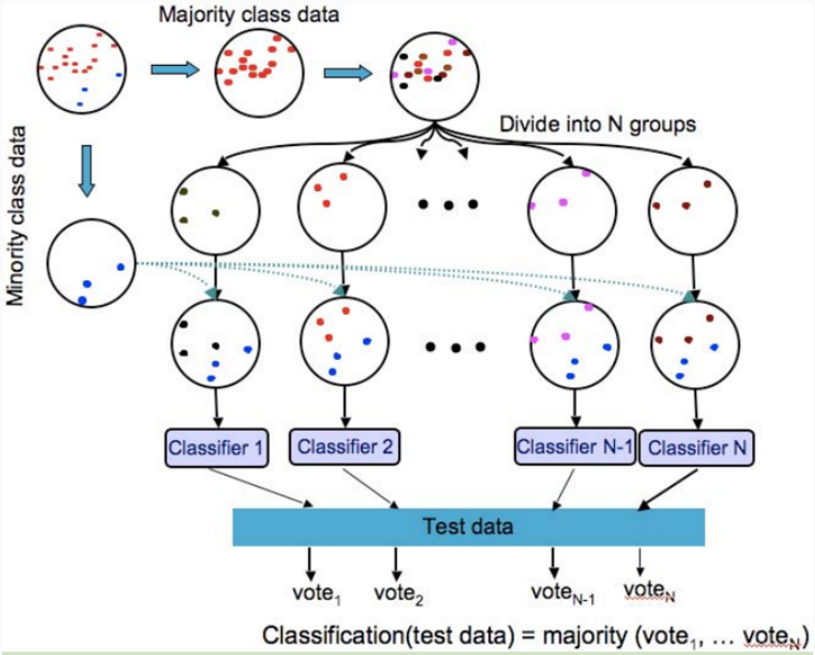
- The same majorty sample maybe sampled repeatedly

X. Y. Liu, J. Wu, and Z. H. Zhou, Exploratory under-sampling for class-imbalance learning, in Proceedings of International Conference on Data Mining, pp. 965969, IEEE Computer Society, 2006.

Informed Undersampling

BalanceCascade

1. Generate $E \subset S_{maj}(s.t.|E| = |S_{min}|)$, and $N = \{E \cup S_{min}\}$
2. Induce $H(n)$
3. Identify N^*_{maj} as samples from N that are correctly classified
4. Remove N^*_{maj} from S_{mag}
5. Repeat (1) and induce $H(n+1)$ until stopping criteria is met



X. Y. Liu, J. Wu, and Z. H. Zhou, Exploratory under-sampling for class-imbalance learning, in Proceedings of International Conference on Data Mining, pp. 965969, IEEE Computer Society, 2006.

Informed Undersampling

BalanceCascade

Pros:

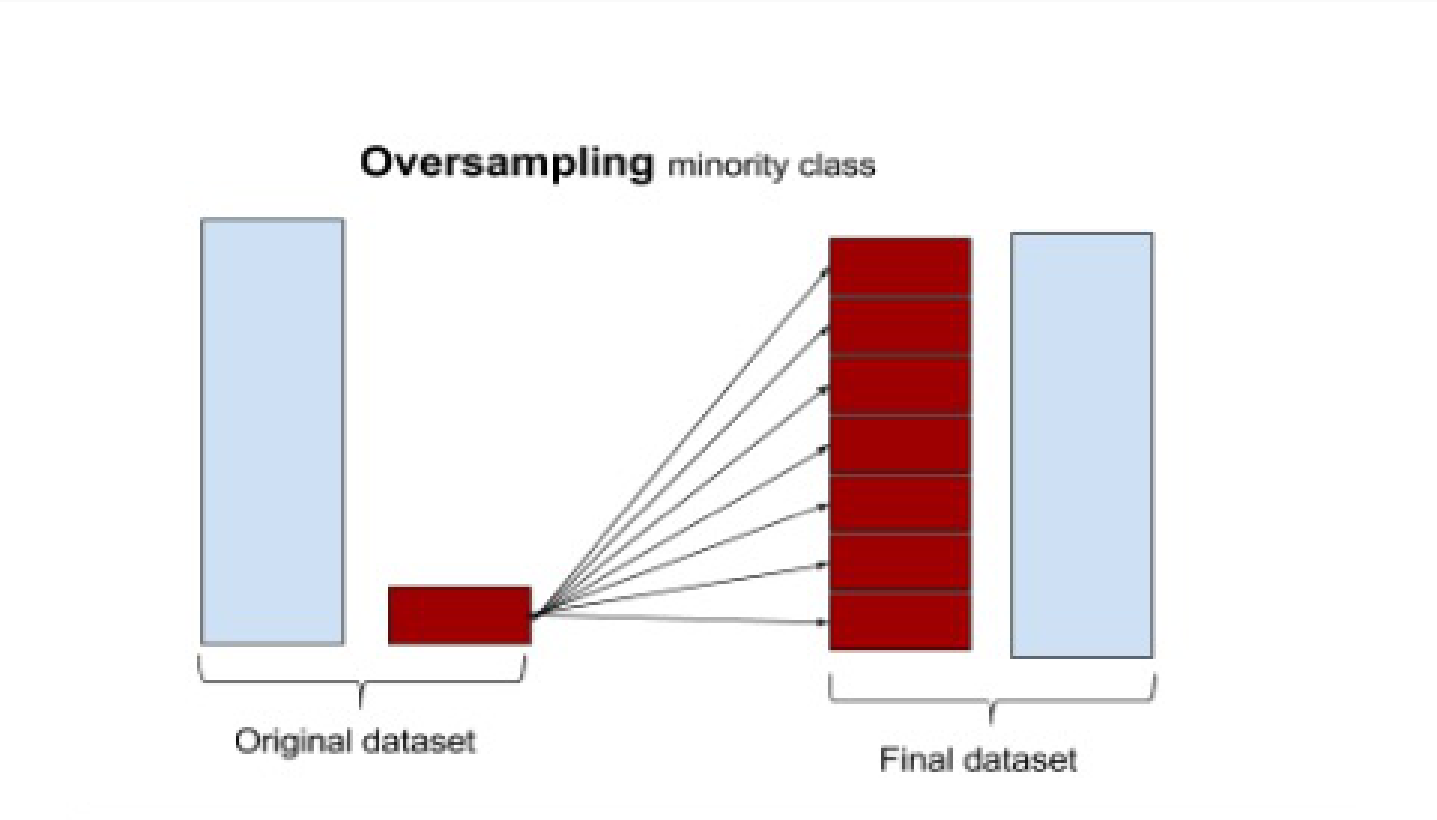
- Save majority samples

Cons:

- Maybe need large amount of classifiers
- Computationally expensive to train them

Synthetic Sampling with Data Generation

Drawback of randomly oversampling



OVERFITTING?

Synthetic Sampling with Data Generation

Drawback of randomly oversampling

Synthetic Sampling with Data Generation

Drawback of randomly oversampling

Synthetic Sampling with Data Generation

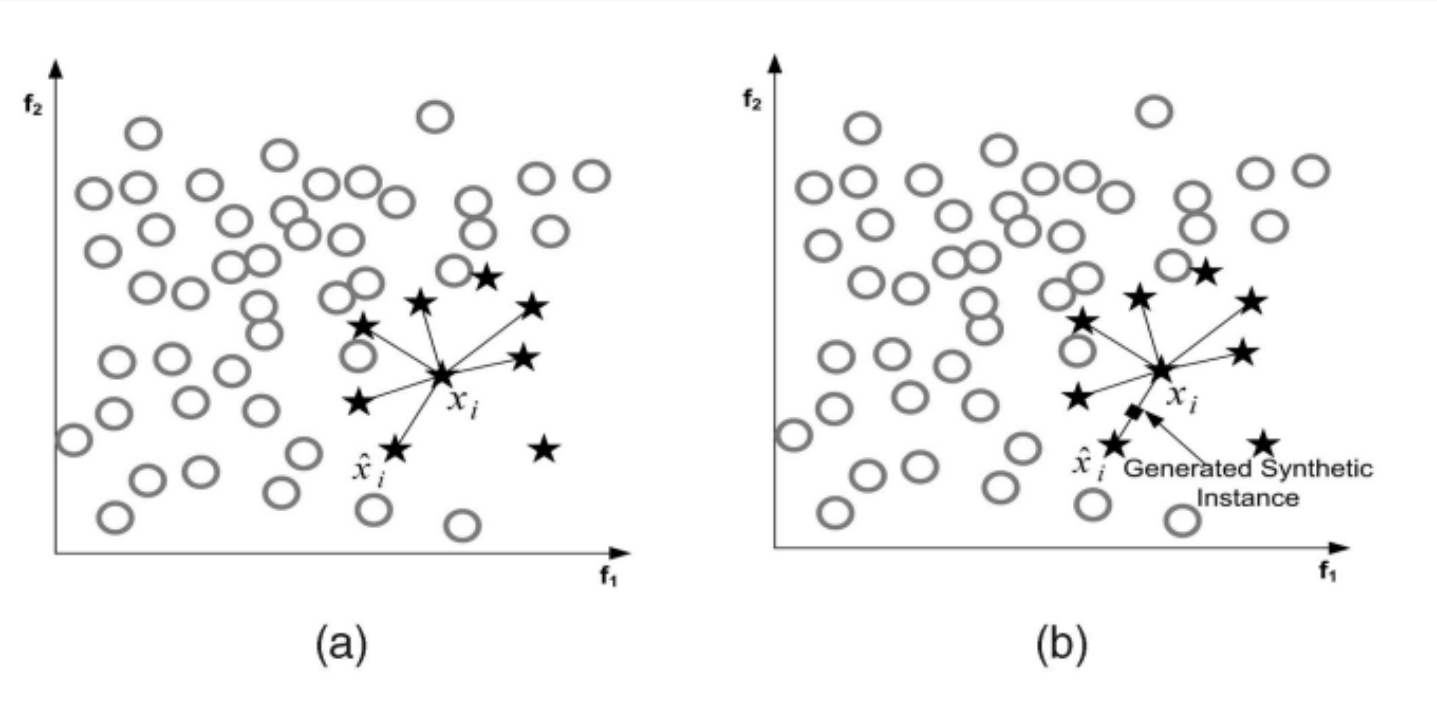
SMOTE(Synthetic Minority Oversampling Technique)

- Creates artificial minority class data using feature space similarities
- For $\forall x_i \in S_{min}$
  1. Randomly choose one of the k nearest neighbor $\hat{x}_i$
  2. Create a new sample $x_{new} = x + \delta * (\hat{x} - x)$, where δ is a uniformly distributed random variable.

N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, vol. 16, pp. 321357, 2002.

Synthetic Sampling with Data Generation

SMOTE



$$x_{new} = x + rand(0, 1) * (\hat{x} - x)$$

N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, vol. 16, pp. 321357, 2002.

Synthetic Sampling with Data Generation

SMOTE

DISADVANTAGE

Overlapping

SOLUTIONS

borderline-SMOTE

adaptive synthetic (ADASYN) sampling

Synthetic Sampling with Data Generation

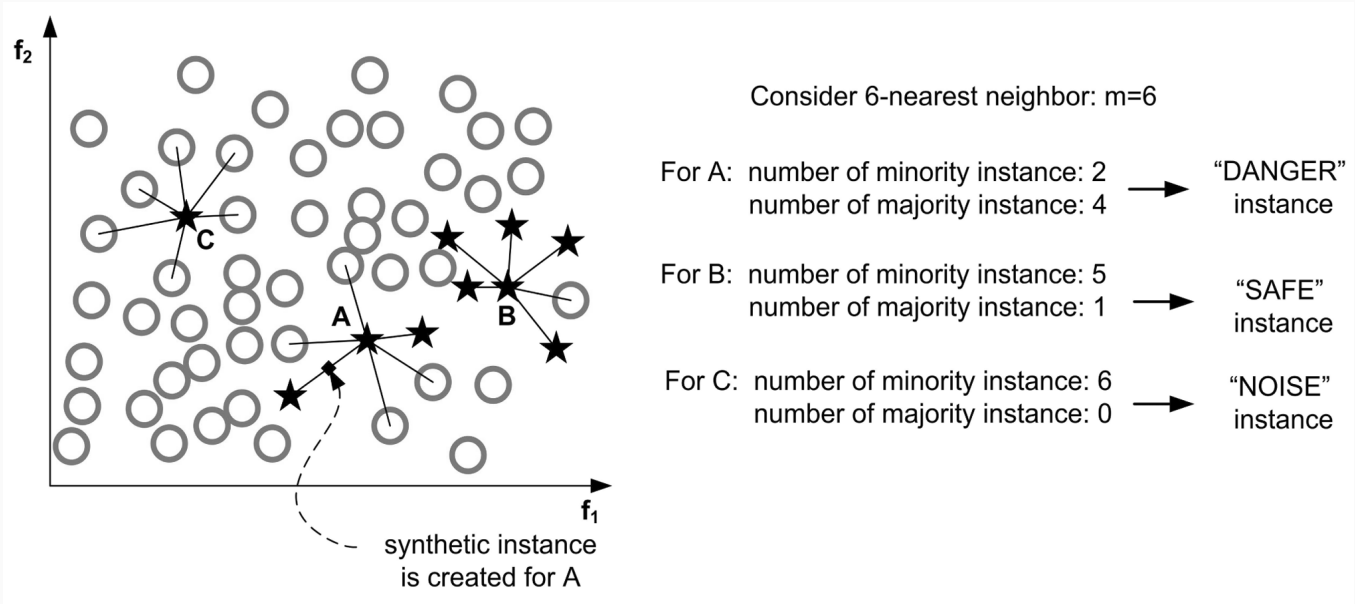
borderline-SMOTE

- Overcomes over generalization in SMOTE algorithm
  1. Determine the set of $m-$nearest neighbors for each $x_i \in S_{min}$, call it $S_{i:m-NN}$
  2. Identify the number of nearest neighbors in majority class, i.e. $|S_{i:m-NN} \cap S_{maj}|$
  3. Select x_i that satisfies: $\frac{m}{2} < |S_{i-mnn} \cap S_{maj}| < m$

H. Han, W. Y. Wang, and B. H. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, in Proceedings of International Conference on Intelligent Computing , Springer, pp. 878887, 2005.

Synthetic Sampling with Data Generation

borderline-SMOTE



DANGER $\quad \frac{m}{2} < \left|S_{i-mnn} \cap S_{maj}\right| < m$

SAFE $\quad \left|S_{i-mnn} \cap S_{maj}\right| \leq \frac{m}{2}$

NOISE $\quad \left|S_{i-mnn} \cap S_{maj}\right| = m$

H. Han, W. Y. Wang, and B. H. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, in Proceedings of International Conference on Intelligent Computing , Springer, pp. 878887, 2005.

Synthetic Sampling with Data Generation

ADASYN

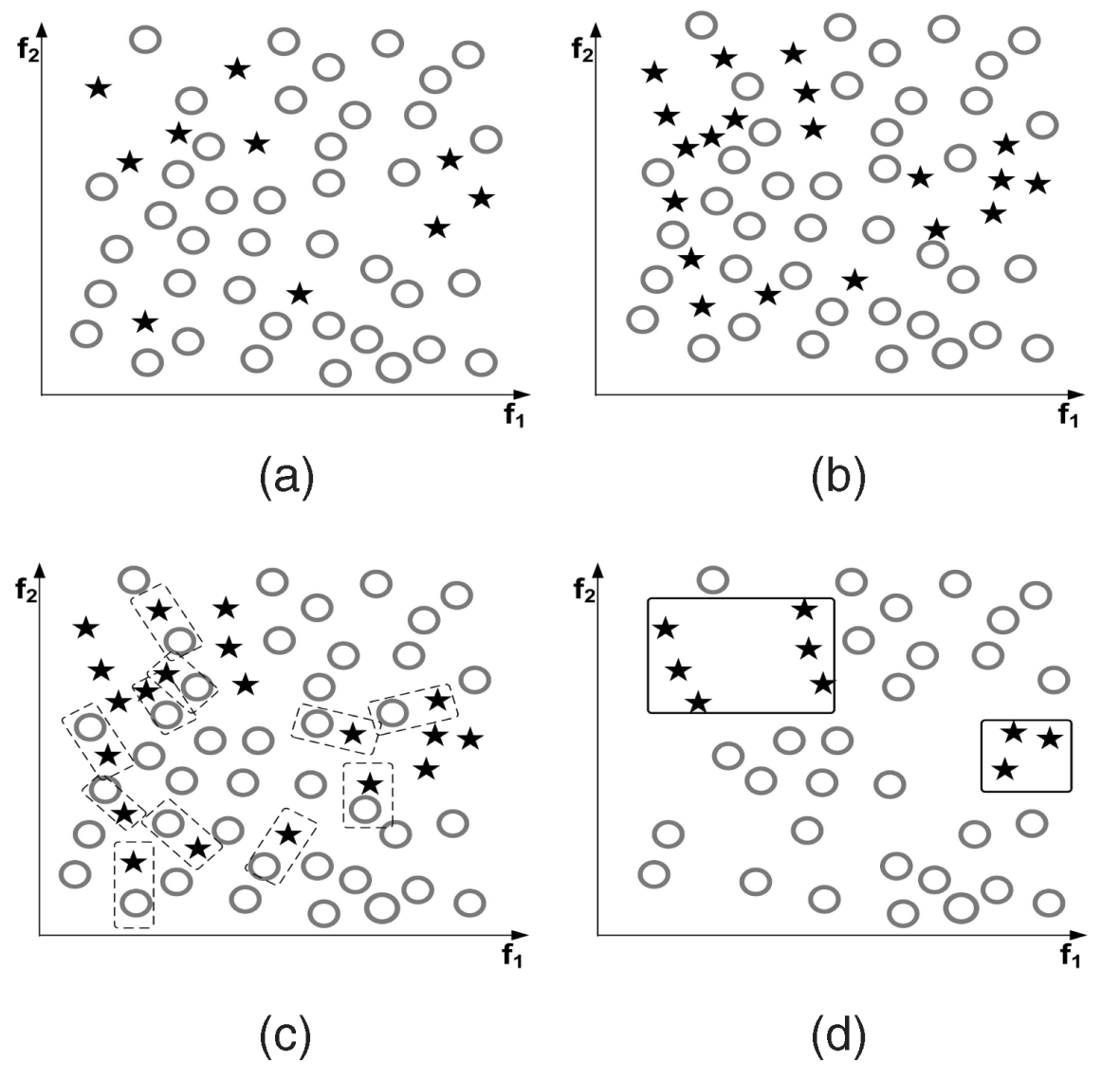
1. Calculate number of synthetic samples
   $G = (|S_{mag}| - |S_{min}|) * \beta$, where $\beta \in [0, 1]$
2. for each $x_i \in S_{min}$, find k-nearest neighbors and calculate ratio $\Gamma_i = \frac{\Delta_i / m}{S_{min}}, i = 1, ..., |S_{min}|$ as a distribution function
3. Identify the number of synthetic samples to be generated for x_i by $g_i = \Gamma_i * G$
4. Generate x_{new} using SMOTE algorithm:
   $x_{new} = x + rand(0, 1) * (\hat{x} - x)$

H. He, Y. Bai, E. A. Garcia, and S. Li, ADASYN: Adaptive synthetic sampling approach for imbalanced learning, in Proceedings of International Joint Conference on Neural Networks, IEEE, pp. 13221328, 2008.

Sampling with Data Cleaning

- Tomek links
  1. Given a pair (x_i, x_j) where $x_i \in S_{min}, x_j \in S_{maj}$, and the distancebetween them as $d(x_i, x_j)$
  2. If there is no instance $x_k s.t. d(x_i, x_k) < d(x_i, x_j)$ or $d(x_j, x_k) < d(x_i, x_j)$, then (x_i, x_j) is called a Tomek link
- Clean up unwanted inter-class overlapping after synthetic sampling

Sampling with Data Cleaning



H. Han, W. Y. Wang, and B. H. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, in Proceedings of International Conference on Intelligent Computing , Springer, pp. 878887, 2005.

Sampling with Data Cleaning

Examples:

- OSS(One-sided selection)
- condensed nearest neithbor and Tomek links(CNN+Tomek links)
- neighborhood cleaning rule(NCL) based on edited nearest neighbor(ENN)
- SMOTE+ENN
- SMOTE+Tomek

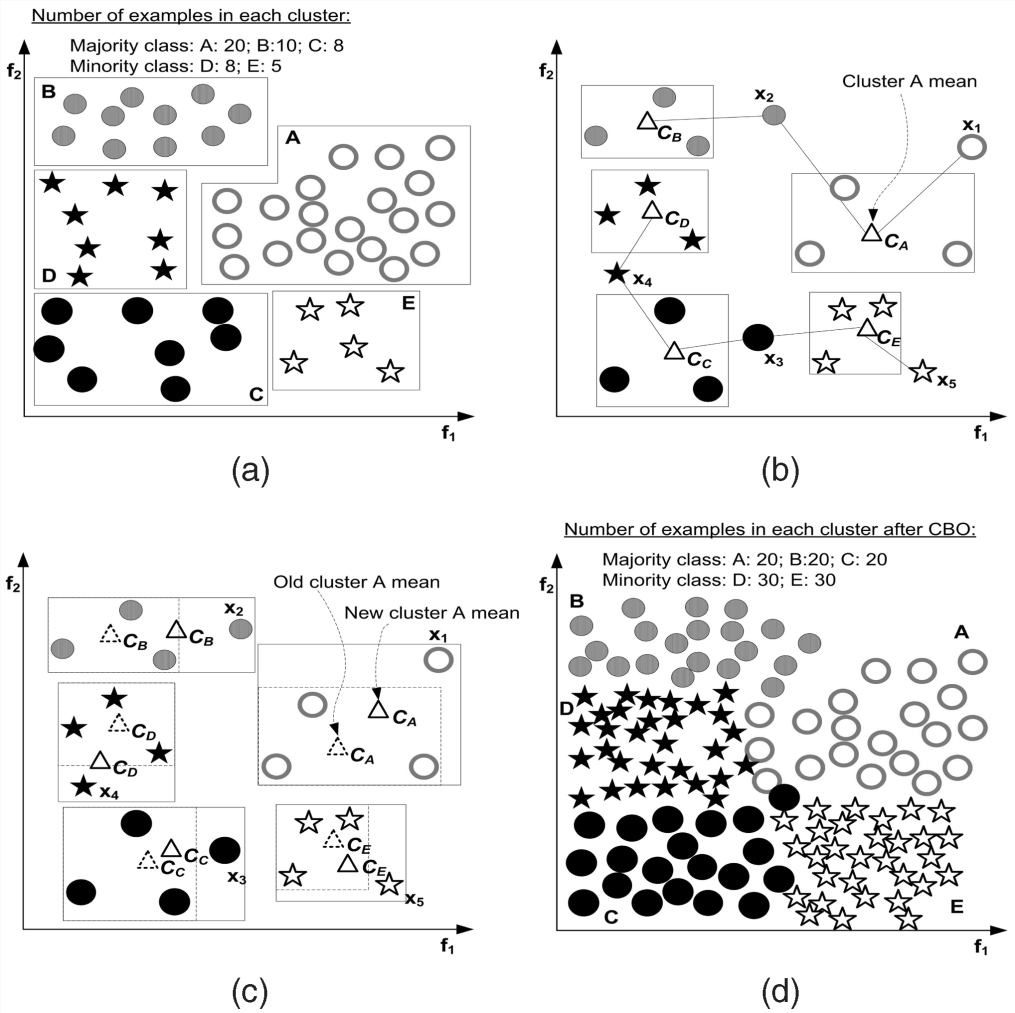
Cluster-based Oversampling

CBO(Cluster-based oversampling)

- For the majority class S_{maj} with m_{maj} clusters
  1. Oversample each cluster $C_{maj:j} \subset S_{maj}, j = 1, ..., m_{maj}$ except the largest $C_{maj:max}$, so that for $\forall j, |C_{maj:j}| = |C_{maj:max}|$
  2. Calculate the number of majority class examples after oversampling as N_{CBO}
- For the minority class S_{min} with m_{min} clusters
  - Oversample each cluster $C_{min:i} \subset S_{min}, i = 1, ..., m_{min}$ to be of the same size $\frac{N_{CBO}}{m_{min}}$, so that for $\forall i, |C_{min:i}| = \frac{N_{CBO}}{m_{min}}$

Cluster-based Oversampling

CBO



H. Han, W. Y. Wang, and B. H. Mao, Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning, in Proceedings of International Conference on Intelligent Computing , Springer, pp. 878887, 2005.

Cluster-based Oversampling

CBO

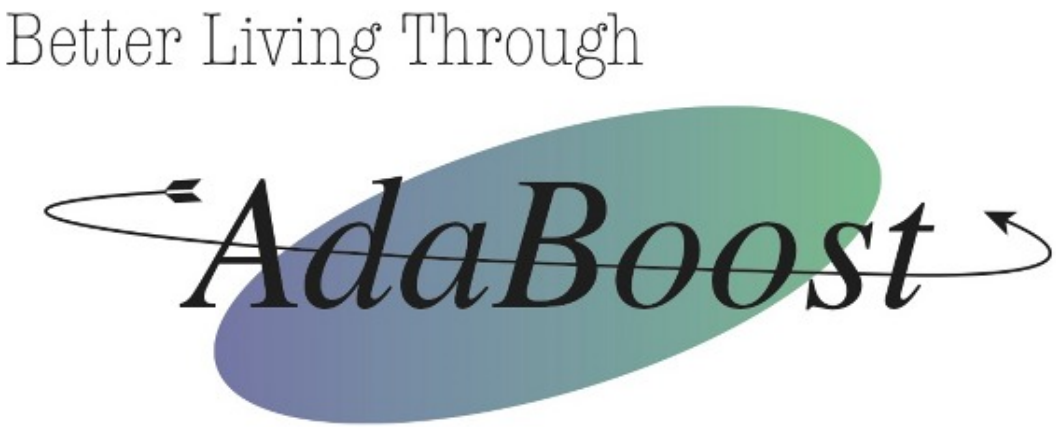
ADVANTAGE

Dealing with overfitting

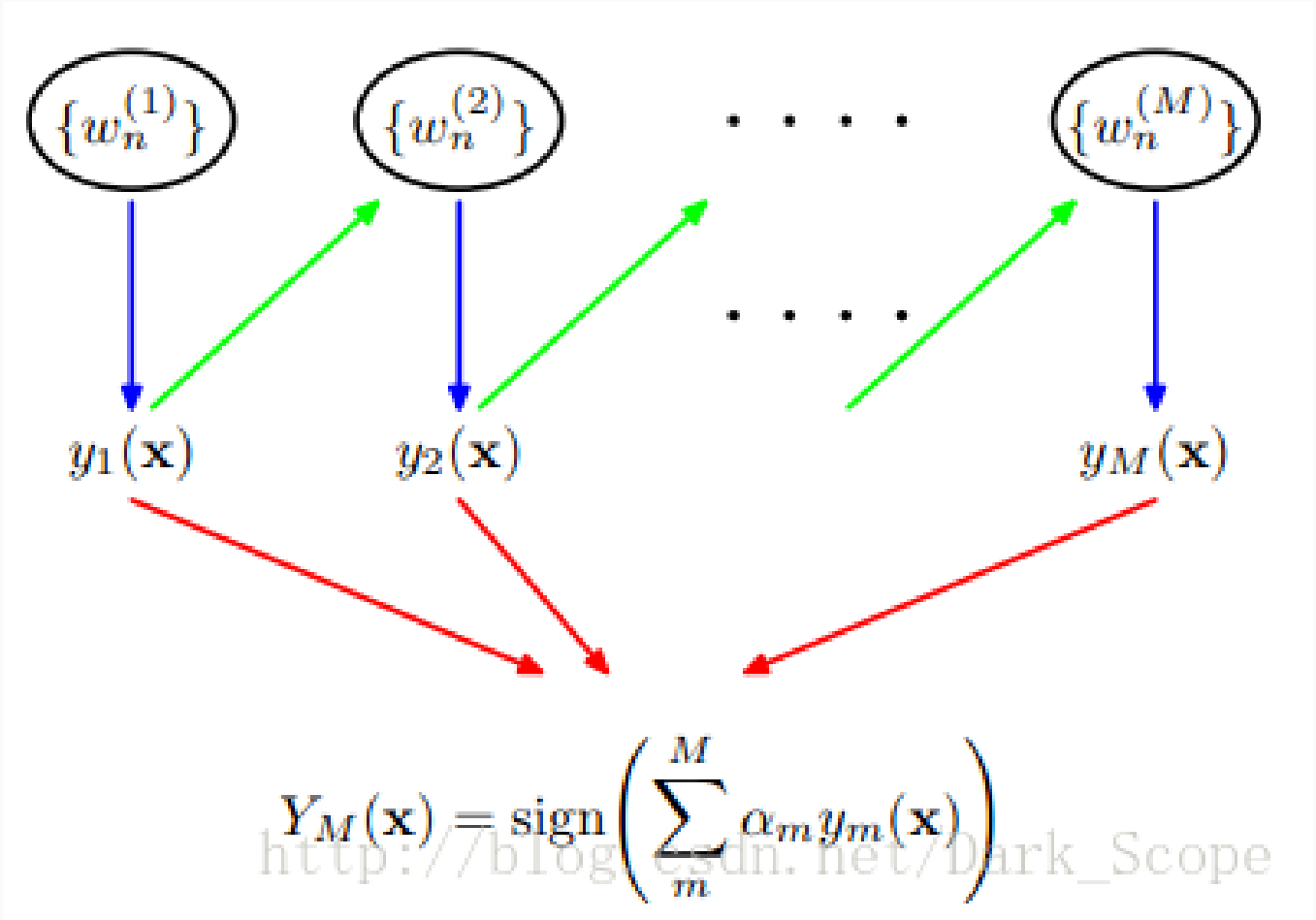
Won't ignore some minority samples

DISADVANTAGE

Computationally expensive

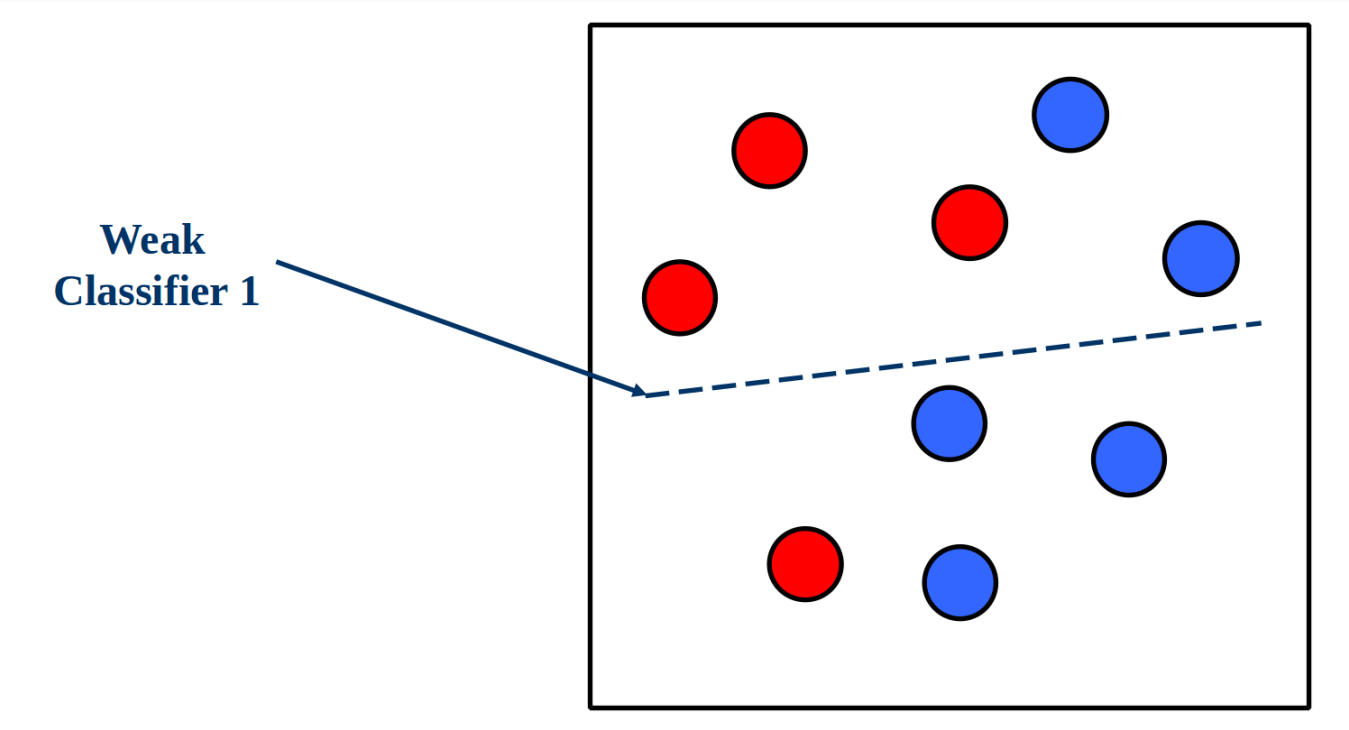

Better Living Through
AdaBoost

Adaboost

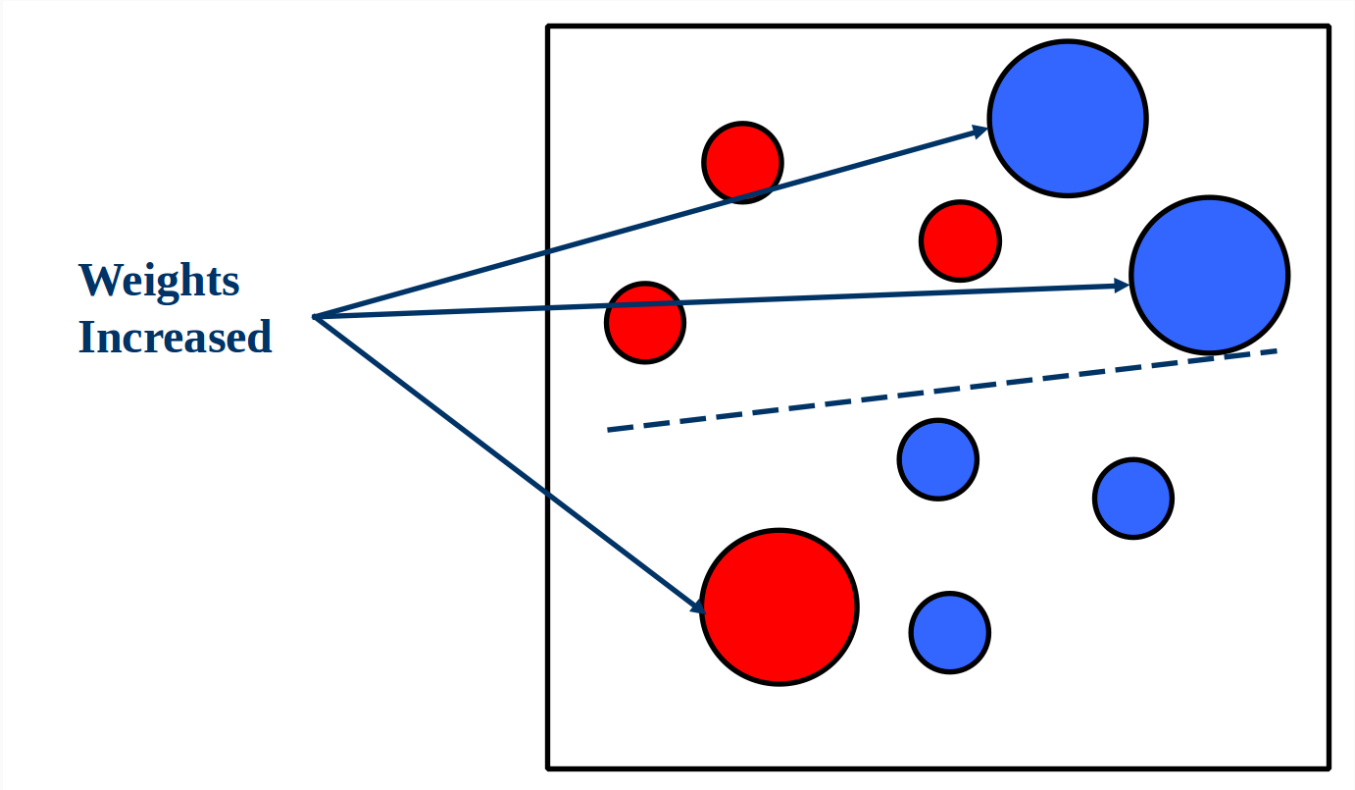

$\{w_n^{(1)}\}$
$\{w_n^{(2)}\}$
$\{w_n^{(M)}\}$
$y_1(\mathbf{x})$
$y_2(\mathbf{x})$
$y_M(\mathbf{x})$
$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$
http://blog.csdn.net/Dark_Scope

Boosting

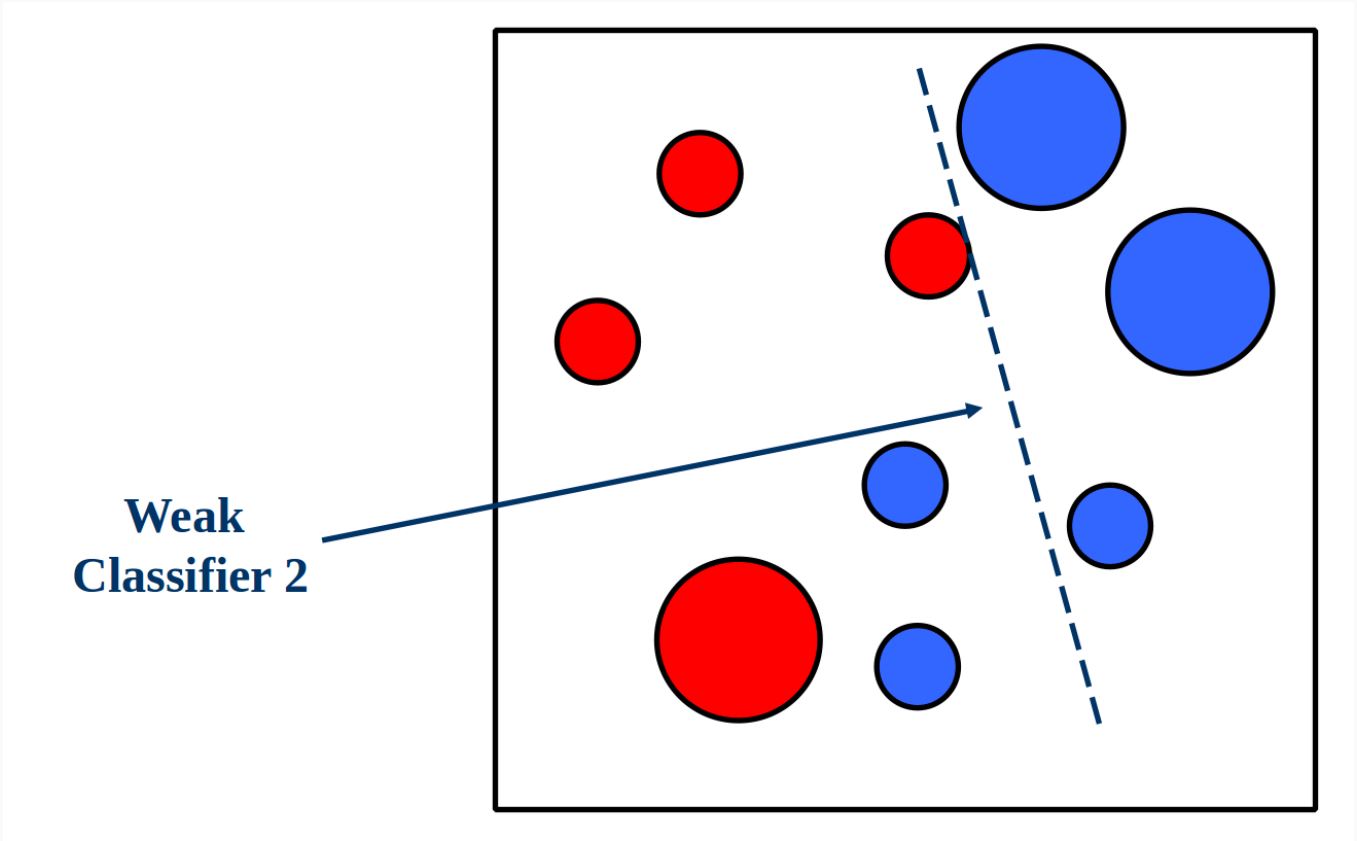
Adaboost



Adaboost

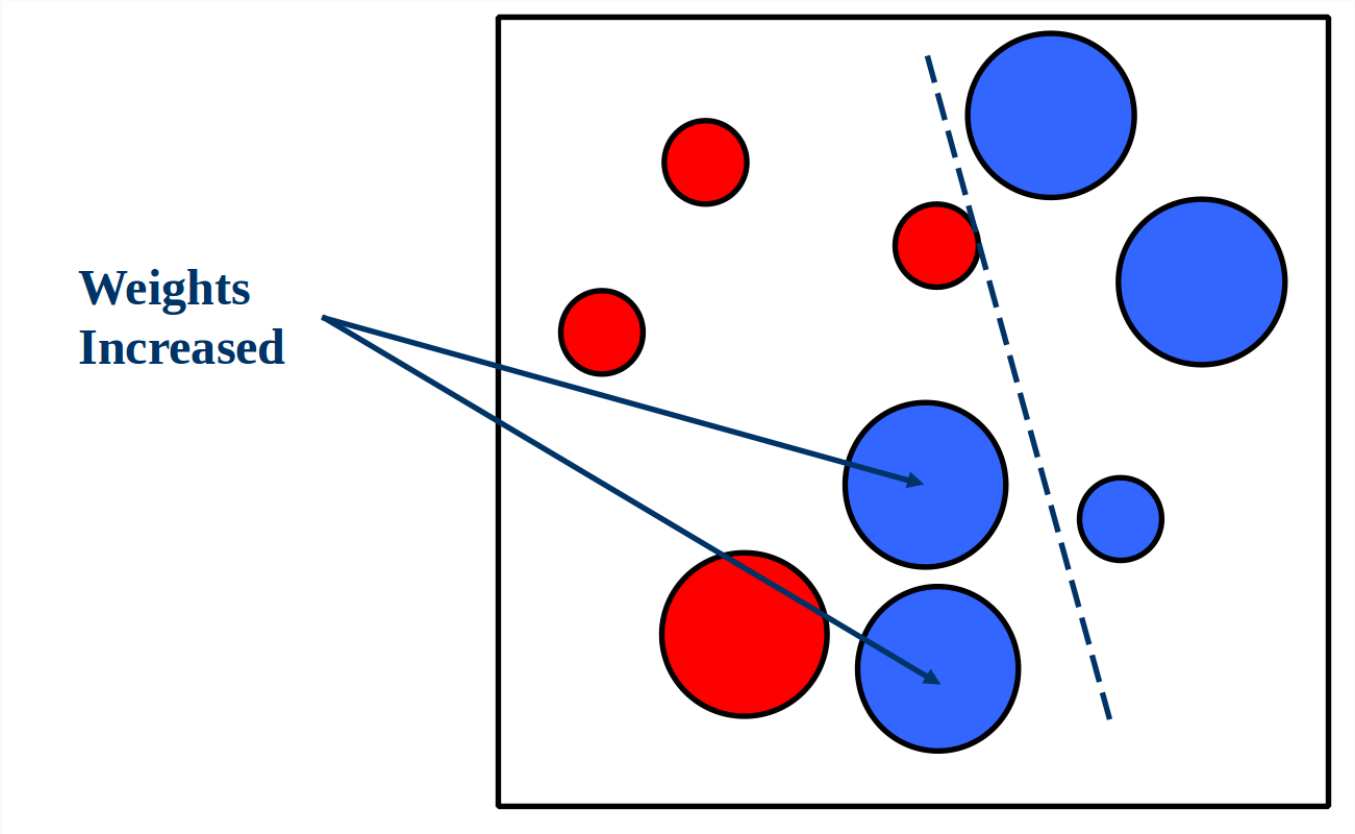


Adaboost

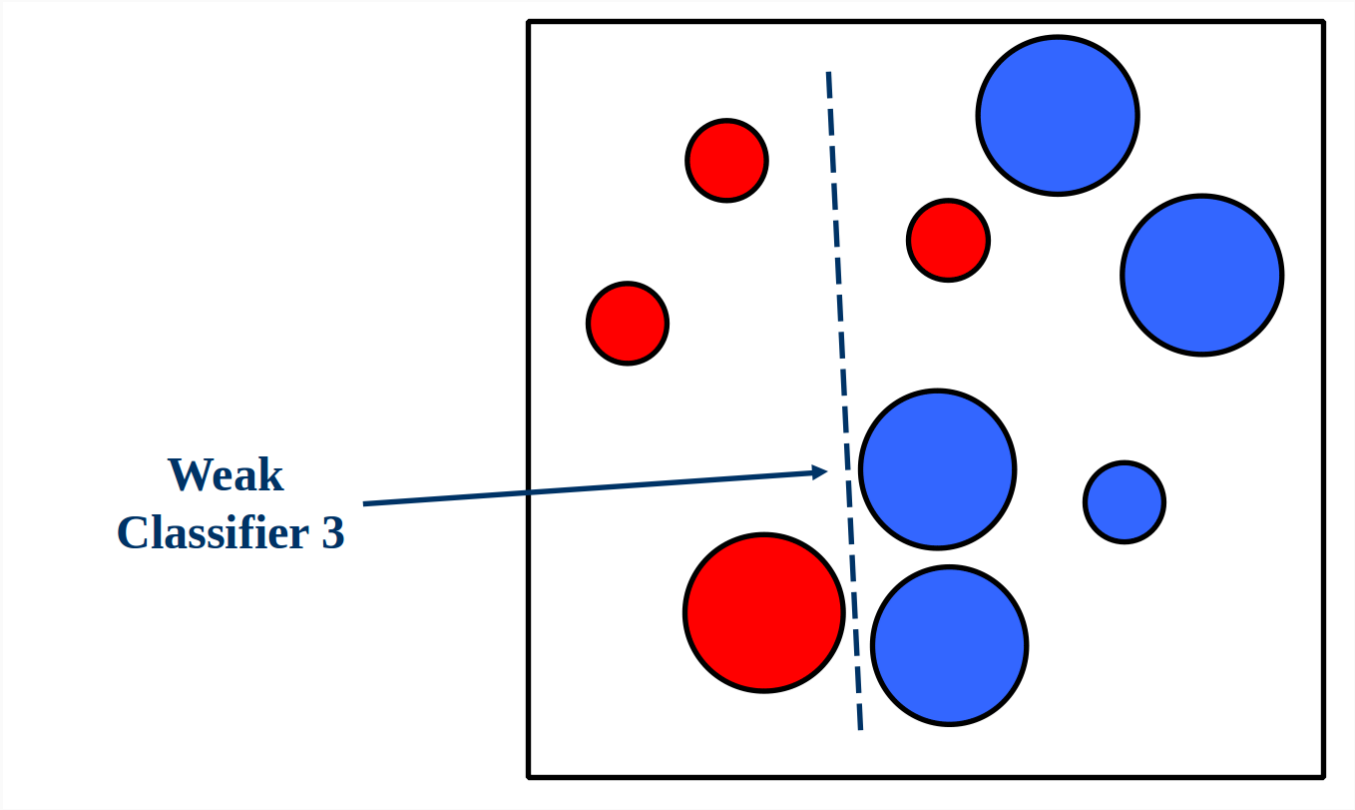


Boosting

Adaboost

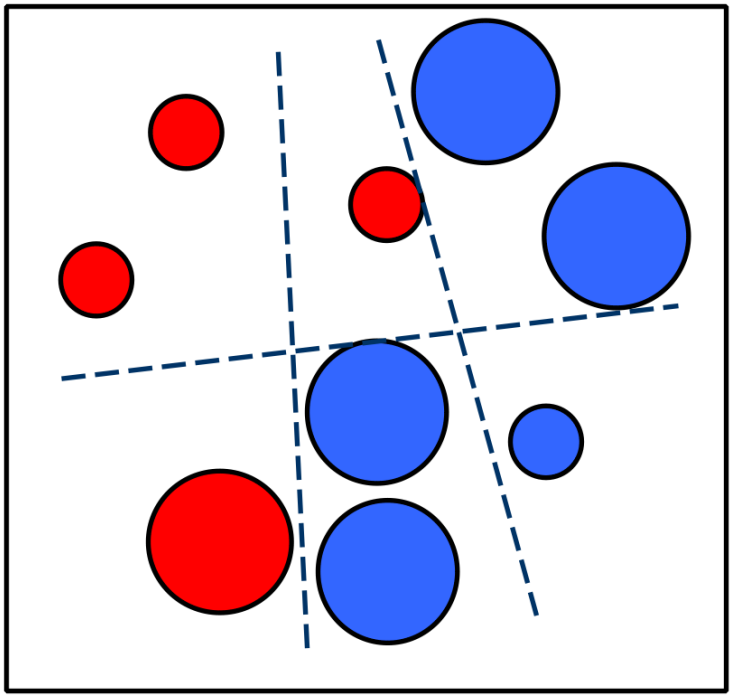


Adaboost



Adaboost

Final classifier is a combination of weak classifiers



Adaboost

1. Initialize the weight of each sample to be $\frac{1}{N}$,where N is the amount of samples
2. for $m = 1, ..., M$:
   - train weak classifier $y_m()$, and it's weighted error function would be: $\epsilon_m = \sum_{n=1}^{N} \omega_n^{(m)} I(y_m(x_n) \neq t_n)$
   - calculate the weight α of this weak classifier:$\alpha_m = \ln \frac{1-\epsilon_m}{\epsilon_m}$
   - update the weight:$w_{m+1,i} = \frac{w_{mi}}{Z_m} exp(-\alpha_m t_i y_m(x_i)), i = 1, 2, ..., N$, where $Z_m = \sum_{i=1}^{N} w_{mi} exp(-\alpha_m t_i y_m(x_i))$
3. gain the final classifier: $Y_M(x) = sign(\sum_{m=1}^{M} \alpha_m y_m(x))$

Integration of Sampling and Boosting

1. SMOTEBoost
   - SMOTE+Adaboost.M2
   - Introduces synthetic sampling at each boosting iteration
2. DataBoost-IM
   - AdaBoost.M1
   - Generate synthetic data of hard-to-learn samples for both majority and minority classes(usually $|E_{maj}| < |E_{min}|$)
3. JOUS-Boost

Cost-Sensitive Learning

Cost-Sensitive Learning

Cost-Sensitive Learning

Cost-Sensitive Learning Framework

- Define the cost of misclassifying a majority to a minority as $C(Min, Maj)$
- Typically $C(Maj, Min) > C(Min, Maj)$
- Minimize the overall cost (usually the Bayes conditional risk) on the training data set

Predict \ True	0	1
0	C_{00}	C_{01}
1	C_{10}	C_{11}

Cost-Sensitive Learning

Implementation of Cost-Sensitive

1. Based on learning models
   - Cost-Sensitive perception machine
   - Cost-Sensitive support vector machine(SVM)
   - Cost-Sensitive decision tree
   - Cost-Sensitive neural network
2. Based on Bayes
   - $$\mathcal{H}(x) = \arg\min_i \left(\sum_{j \in \{-,+\}} P(j|x) C(i,j) \right)$$
3. Adjust the weight in preprocessing

Cost-Sensitive Decision Trees

1. Cost-Sensitive adjustments for the decision threshold
   - The final decision threshold shall yield the most dominant point on the ROC curve
2. Cost-sensitive considerations for split criteria
   - The impurity function shall be insensitive to unequal costs
3. Cost-sensitive pruning schemes
   - The probability estimate at each node needs improvement to reduce removal of leaves describing the minority concept
   - Laplace smoothing method and Laplace pruning techniques

Cost-Sensitive Dataspace Weighting with Adaptive Boosting

Iteratively update the distribution function D_t of the training data according to error of current hypothesis h_t and cost factor C_i

- Weight updating parameter $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
- Error of hypothesis $h_t : \varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$

Cost-Sensitive Dataspace Weighting with Adaptive Boosting

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

Given $D_t, h_t, C_i, \alpha_t,$ andε_t

- AdaC1: $D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t C_i h_t(x_i)y_i)}{Z_t}$
- AdaC2: $C_i D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t h_t(x_i)y_i)}{Z_t}$
- AdaC3: $C_i D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t C_i h_t(x_i)y_i)}{Z_t}$
- AdaCost: $D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t h_t(x_i)y_i)\beta_i}{Z_t}$

  $\beta_i = \beta(sign(y_i, h_t(x_i)), C_i)$

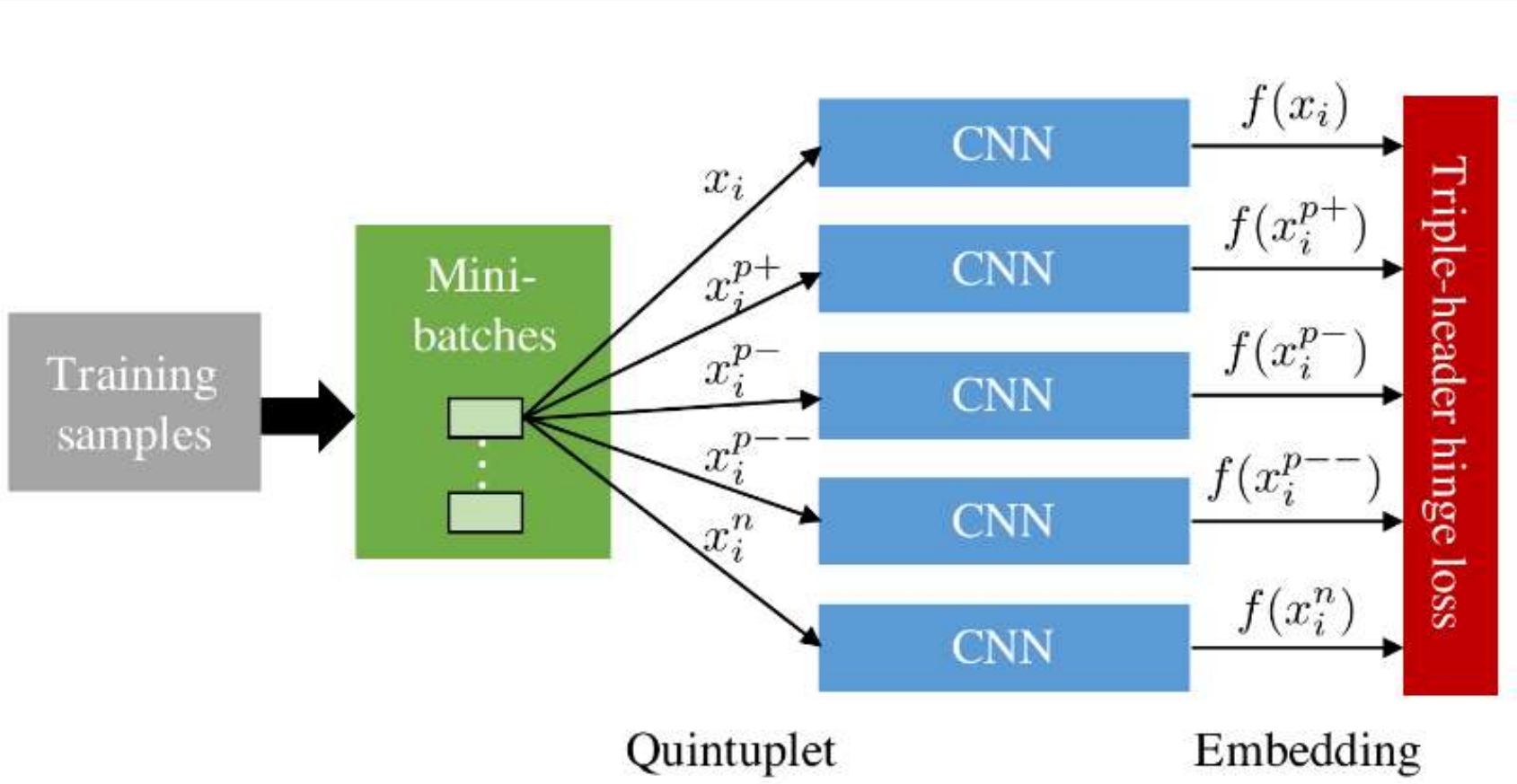
Cost-Sensitive Learning

Four ways of applying cost sensitivity in neural networks

Modifying probability estimate of outputs	Altering outputs directly	Modify learning rate	Replacing error-minimizing function
• *Applied only at testing stage* • *Maintain original neural networks*	• *Bias neural networks during training to focus on expensive class*	• *Set η higher for costly examples and lower for low-cost examples*	• *Use expected cost minimization function instead*

Cost-Sensitive Learning

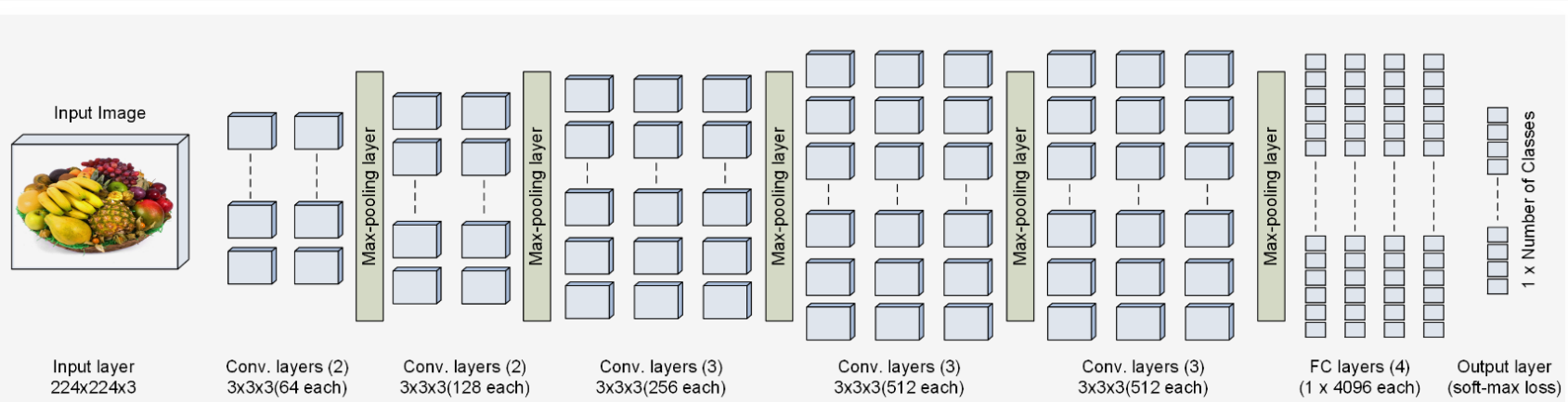
Cost sensitivity in neural networks



Huang C, Li Y, Change Loy C, et al. Learning deep representation for imbalanced classification[C]IEEE Conference on CVPR. 2016: 5375-5384.

Cost-Sensitive Learning

Cost sensitivity in neural networks



Khan S H, Bennamoun M, Sohel F, et al. Cost sensitive learning of deep feature representations from imbalanced data[J]. preprint arXiv:1508.03422, 2015.

Evaluation Criteria

Evaluation Criteria

		True class: p	True class: n
Hypothesis output	Y	TP (True Positives)	FP (False Positives)
	N	FN (False Negatives)	TN (True Negatives)
Column counts:		P_C	N_C

$$Precision: p = \frac{TP}{TP + FP}$$

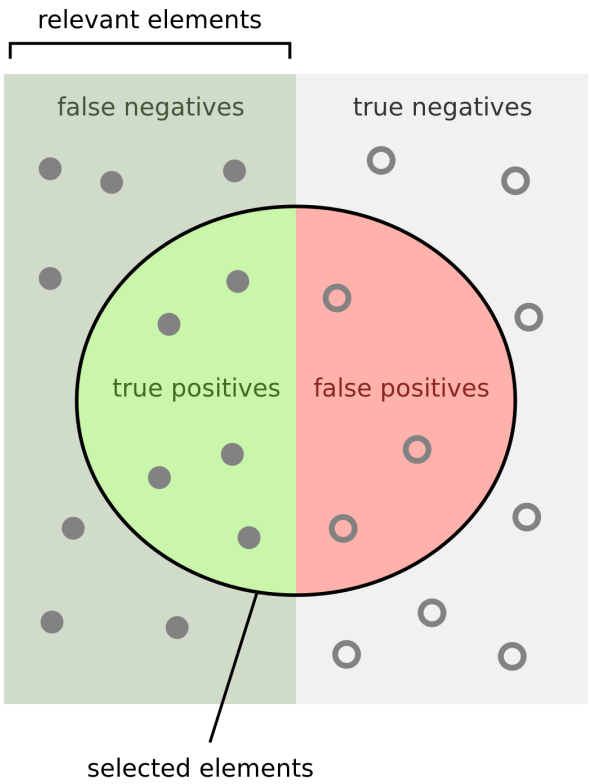
$$Recal: r = \frac{TP}{TP + FN}$$

$$acc_+ = \frac{TP}{TP + FN}$$

$$acc_- = \frac{TN}{TN + FP}$$

Assessment Metrics

Assessment Metrics



How many selected items are relevant?

Precision =

How many relevant items are selected?

Recall =

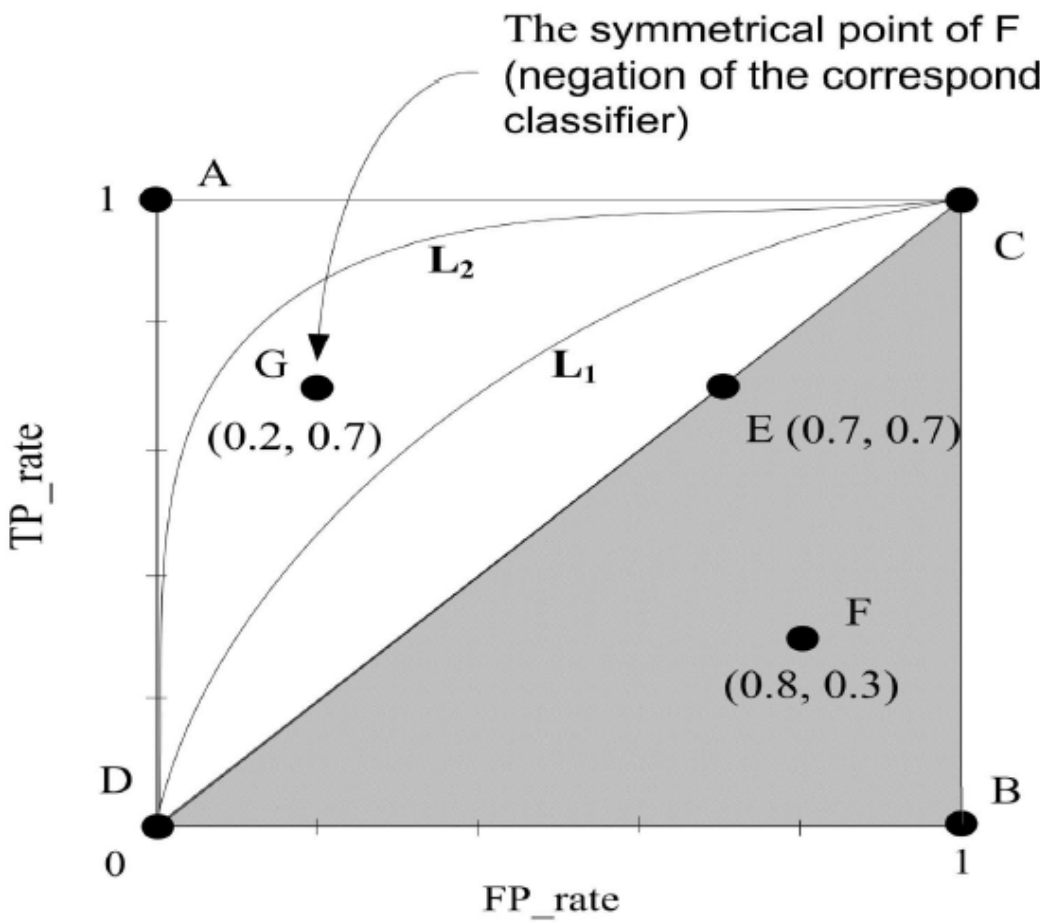
$$G - means = \sqrt{acc_+ * acc_-}$$

$$F - measure = \frac{1}{\frac{1}{2}(\frac{1}{p} + \frac{1}{r})} = \frac{2pr}{p + r}$$

Receive Operating Characteristics(ROC) curves

$$FP\_rate = \frac{FP}{FP + TN}$$

$$TP\_rate = \frac{TP}{TP + FN}$$



Area under the curve(AUC)

Thank you!