Learning from highly imbalanced data

Jordan Fréry

PhD student at the laboratory Hubert Curien and at the company Atos Worldline

Outline

- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Research vs real life

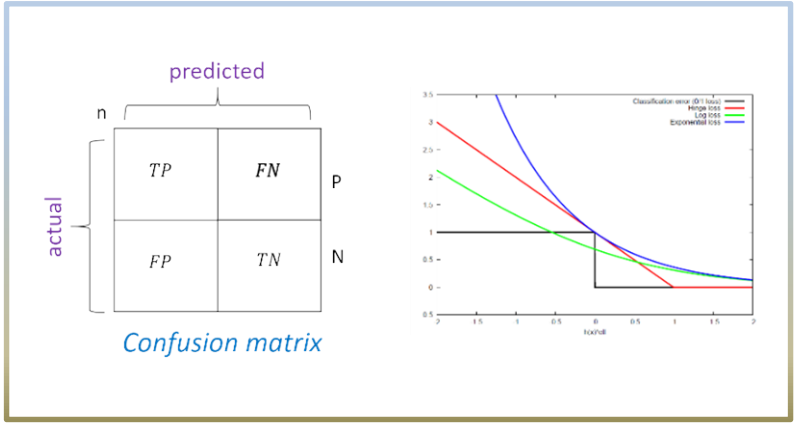
- A dataset in research is considered unbalanced when the minority class is between 10% and 20%.
- In reality, datasets get far more imbalanced than this :
  - Credit card fraud datasets < 0.2% for the minority class
  - Medical screening, e.g. HIV prevalence in the USA is 0.4%
  - Disk drive failure < 1%

Research vs real life

- Datasets are much bigger and noisy in real life
- It is very unlikely that you will find a balanced dataset

Why is it a problem?

- Loss function $L(f(x), y)$: agreement between prediction $f(x)$ and desired output y.
- Common loss functions are accuracy based
- Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$
- For a dataset with 0.2% positives, the model achieves 99.8% accuracy by just claffisying the negative class correctly.



Outline

- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Metrics

Accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$

In fact when #positives << #negatives we would rather look at different metrics such as:

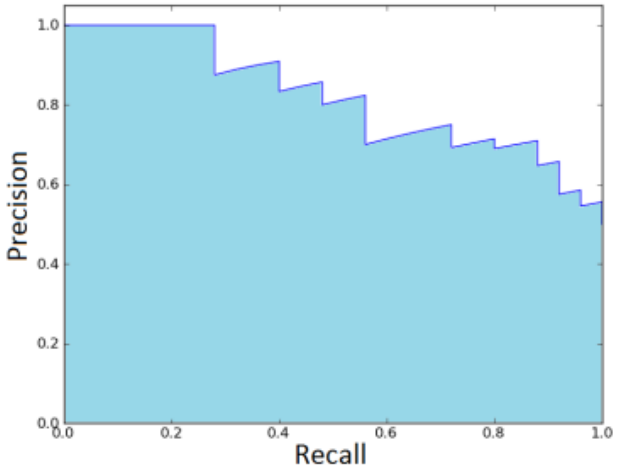
- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- $F_\beta - score = \frac{(1+\beta^2)*precision*recall}{(\beta^2*precision)+recall}$

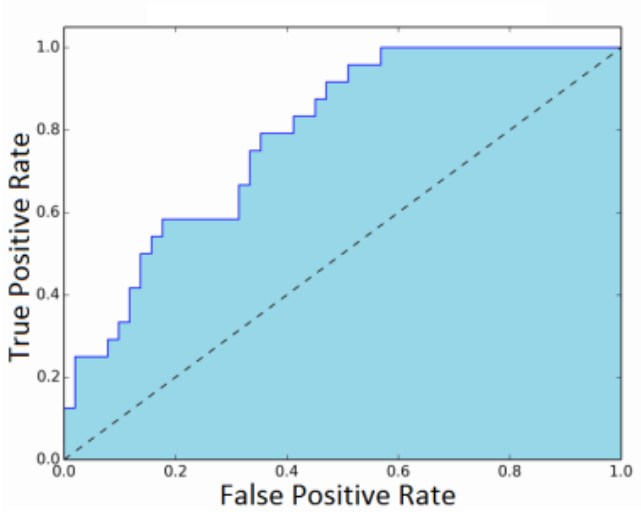
These metrics are highly dependent on the decision threshold defined for the classifier

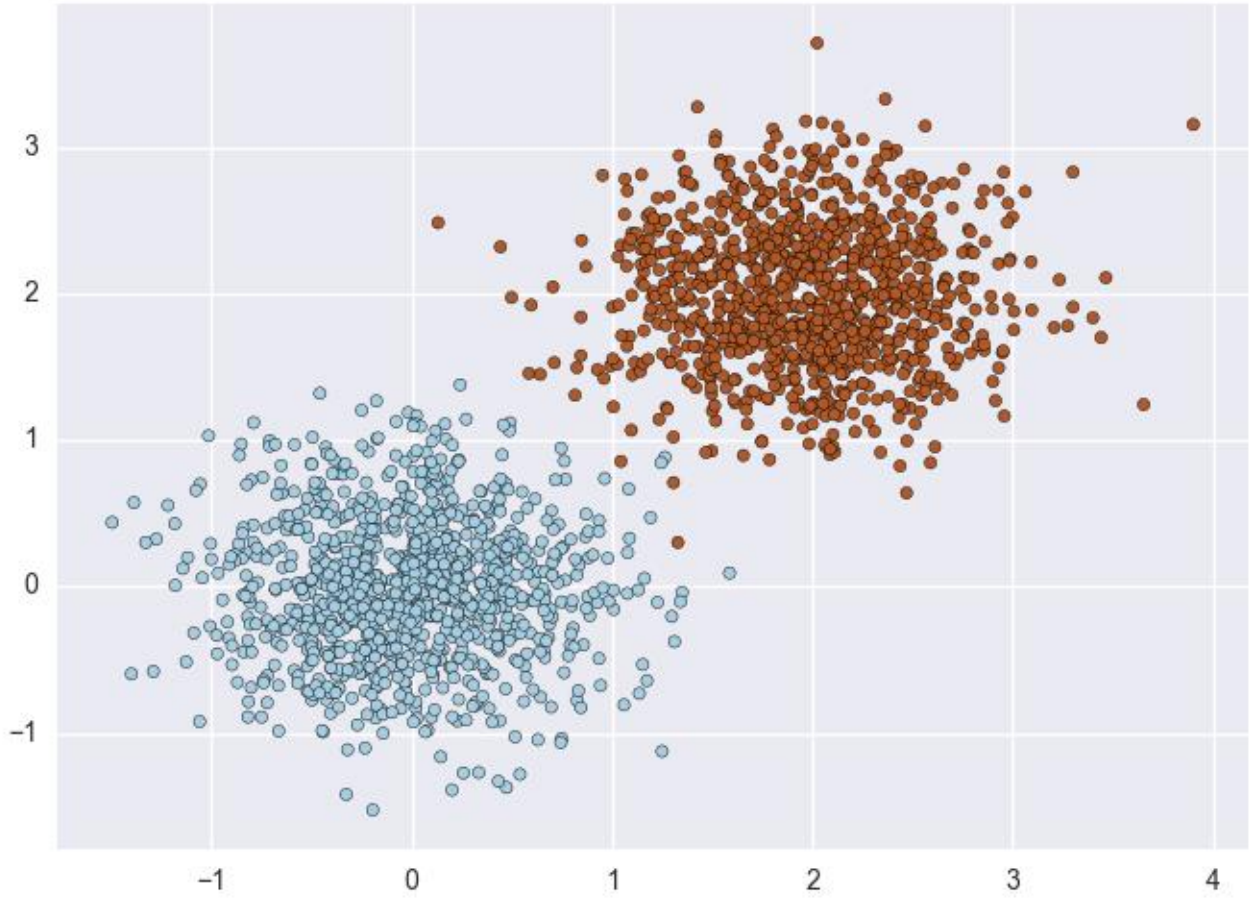
Metrics

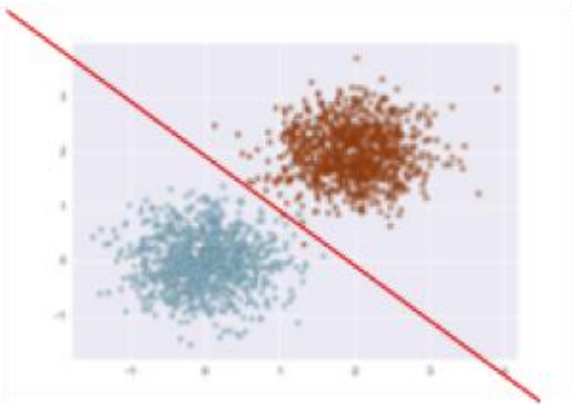
$$AUCAP = \frac{1}{P}\sum_{i=1}^{P} precision@k$$

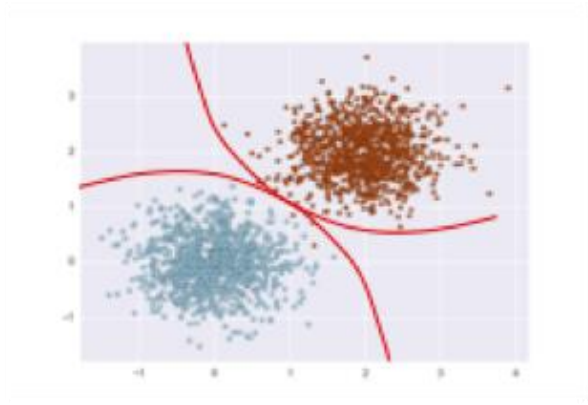
$$AUCROC = \frac{1}{PN}\sum_{i=1}^{P}\sum_{j=1}^{N} \mathrm{I}\left(f(x_i) > f(x_j)\right)$$

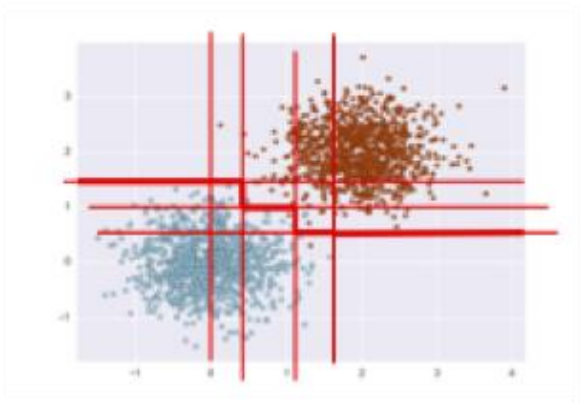


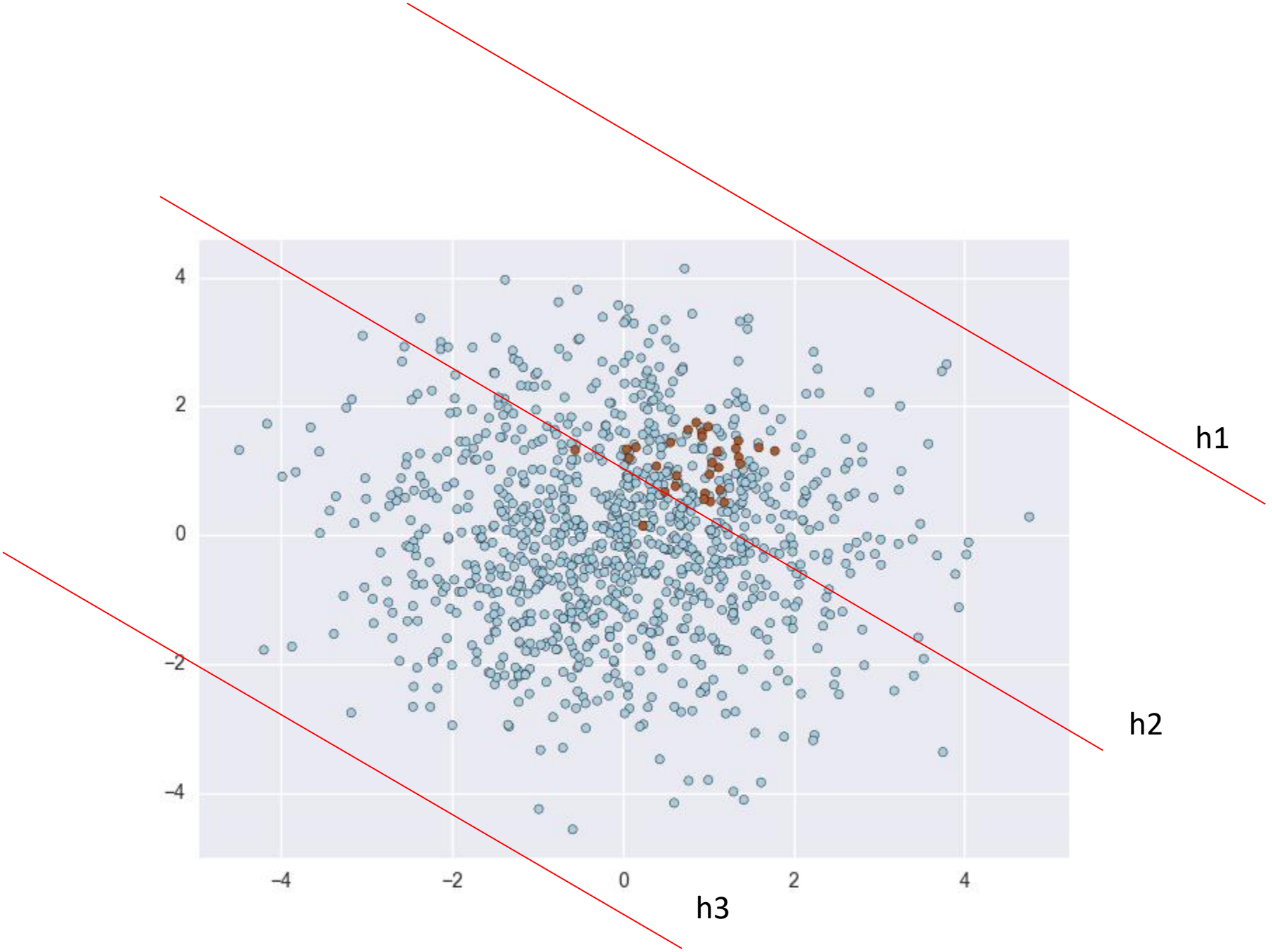



3
2
1
0
−1
−1
0
1
2
3
4








h1
h2
h3
4
2
0
−2
−4
−4
−2
0
2
4

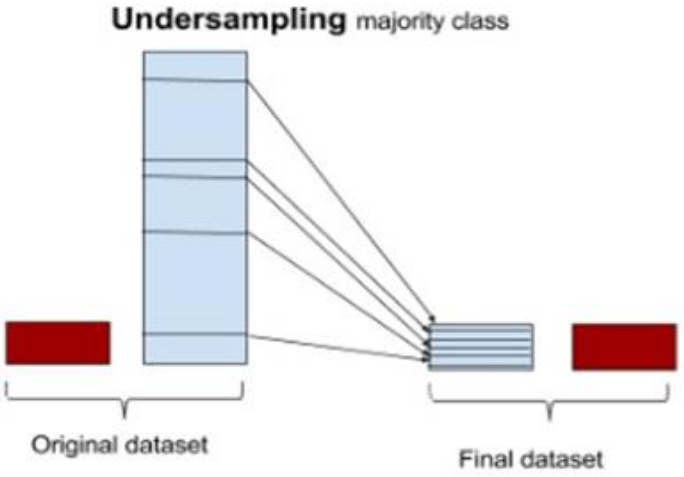
Outline

- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Re-sampling the training set

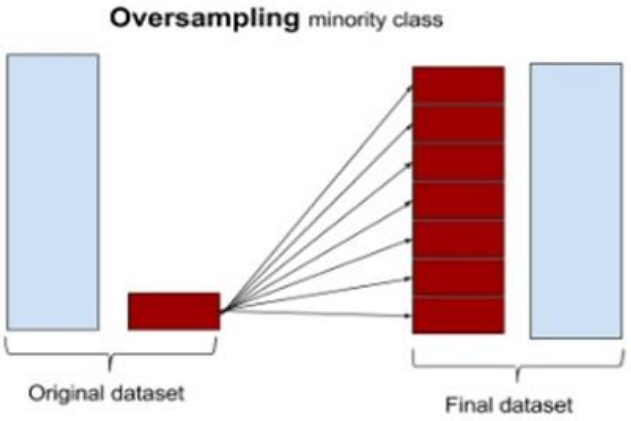
- Undersampling:

Randomly downsample the majority class



- Oversampling:

Randomly replicates minority instances



Impact on the classifier

- Undersampling:

Pros:

  - Faster training (reduce training set size)
  - Add more weight for the minority class

Cons:

  - Potential loss of information

- Oversampling:

Pros:

  - Add more weight for the minority class

Cons:

  - Slows training (increases training set size)
  - Reduce the variance of the minority class (model is prone to overfitting)

Tomek links

Idea

Instead of removing randomly examples from the majority class, remove those that are the closer to the point in the minority class.

For each minority example x

- Find the T-links such that $d(x,y)<d(x,z)$ or $d(x,y)<d(y,z)$, where y is any example from the majority class and z is any other instance than x and y.
- Remove the T-links

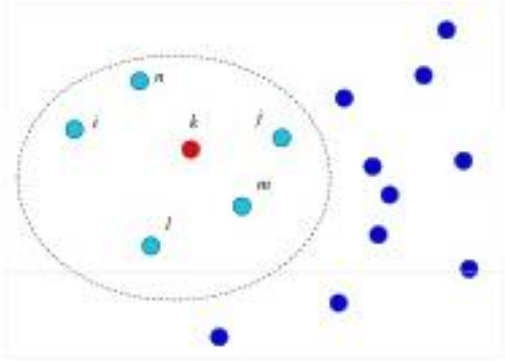
SMOTE

Idea

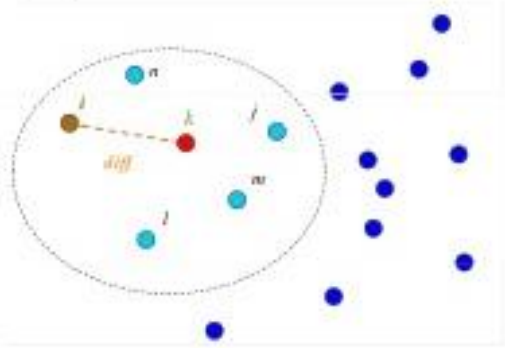
Include more variance in the new examples

For each minority Sample

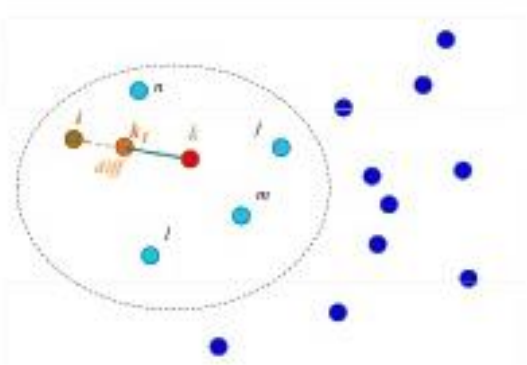
- Find its k-nearest minority neighbors
- Randomly select j of these neighbors
- Randomly generate synthetic samples along the line joining the minority sample and its j selected neighbours
- (j depends on the amount of oversampling desired)



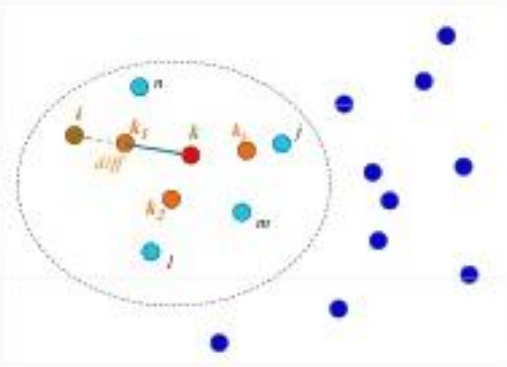
1. For each minority example k compute nearest minority class examples (i, j, l, n, m)



2. Randomly choose an example out of 5 closest points



3. Synthetically generate event k_1, such that k_1 lies between k and i



4. Dataset after applying SMOTE 3 times

SMOTE

Pros:

- Adds diversity among the new examples

Cons:

- Makes the learning process longer
- Cautious when in presence of highly imbalanced dataset -> how do we preserve diversity ?

Note that there is also the version where you add new example within the hypercube rather than the line between two examples. (weka vs imbalance-learn)

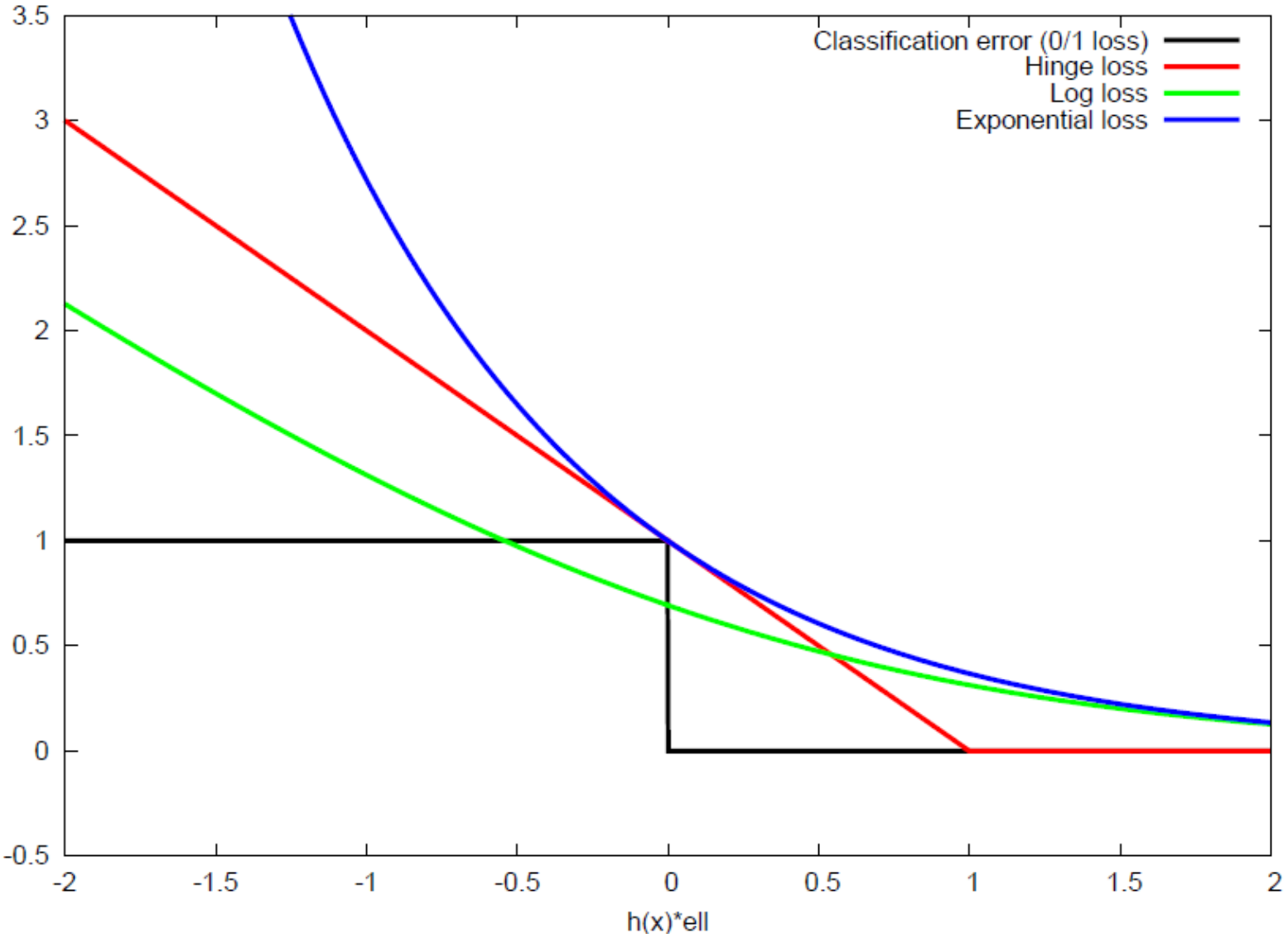
And many more…

- SMOTE + Random Undersampling
- Borderline SMOTE
- Condensed Nearest Neighbor (CNN)
- Edited Nearest Neighbor (ENN)
- SMOTE + CNN
- SMOTE + ENN
- SMOTE + Tomek links
- …

Outline

- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Common loss functions



Example of different loss functions

Adapted objective function?

- Approximations of the 0/1 loss are based on the accuracy

- Common loss functions are very practical:
  - Differentiable and continuous
  - Convex

Having these properties in a new loss is difficult…

- One approach is to use cost sensitive learning:
  - Use a practical loss function weighted by arbitrary costs

Cost based learning

One need to define : C_{TP}, C_{TN}, C_{FP}, C_{FN} for each cell in the confusion matrix

Expected cost :

$R(i|x) = \sum_j P(j|x)C(i,j)$, with j the true label and i the predicted class

x is positive if and only if:

$$P(0|x)C_{FP} + P(1|x)C_{TP} \leq P(0|x)C_{TN} + P(1|x)C_{FN}$$

Cost based learning

Cost based learning with financial cost:

- We can define: C_{TP_i}, C_{TN_i}, C_{FP_i}, C_{FN_i} , the costs of a specific example (e.g. based on the amount of a transaction)

Cost based learning

Pros:

- No loss of information
- Can be easily adapted for specific costs (i.e. buisness related)

Cons:

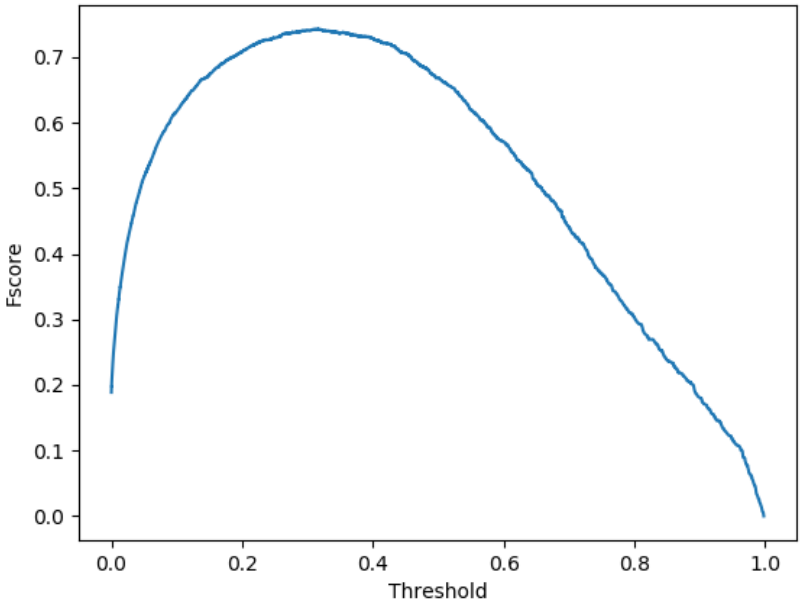
- Hard to set the costs
- The learning algorithm must be able to handle the costs

Outline

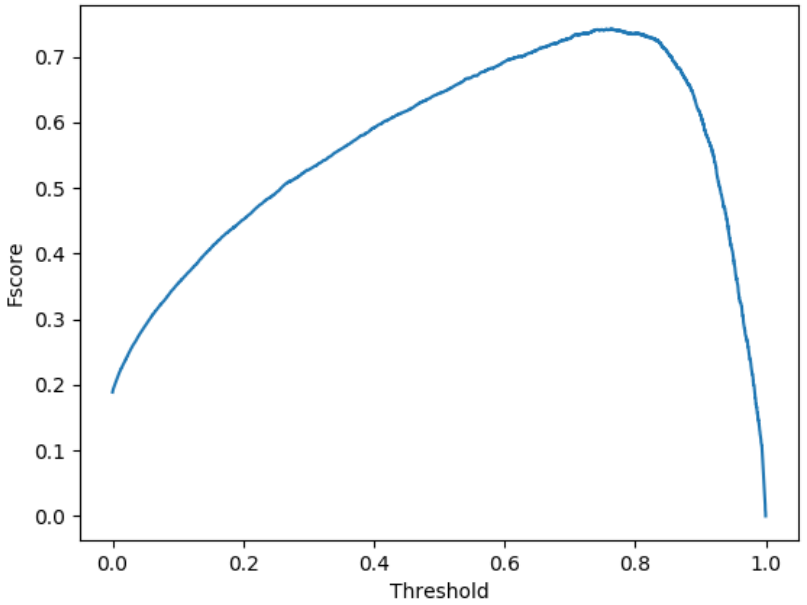
- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Posterior probability problem

- Need of thresholding



Without sampling



With sampling

Posterior probability problem

- Only a problem for classifier (metrics such as ROC or AP do not depend on the threshold)

- Calibrate the posterior probability
  - Using a validation set
  - During the learning process?

Outline

- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Ensemble methods

- Definition : collection of classifiers that are minor variants of the same classifier
- Key point : create diverse classifiers to better generalize
- Tends to work better on imbalanced dataset than basic model such as logistic regression.

Ensemble methods

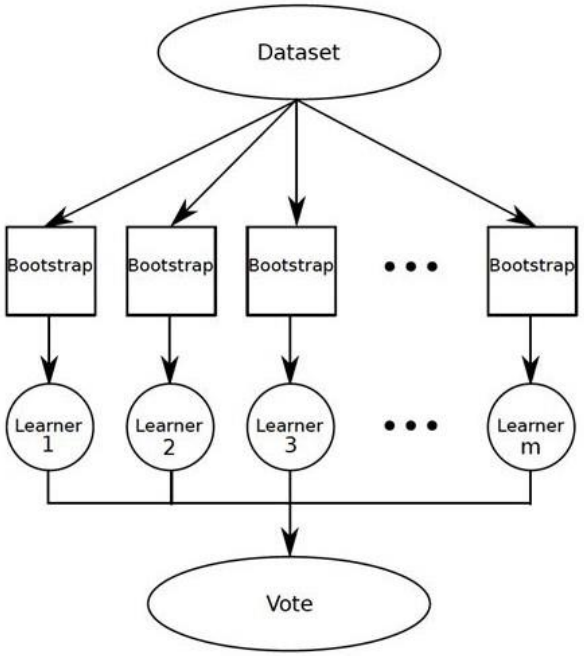
Bagging:

Goal : reduce the variance of the model (e.g. decision tree)

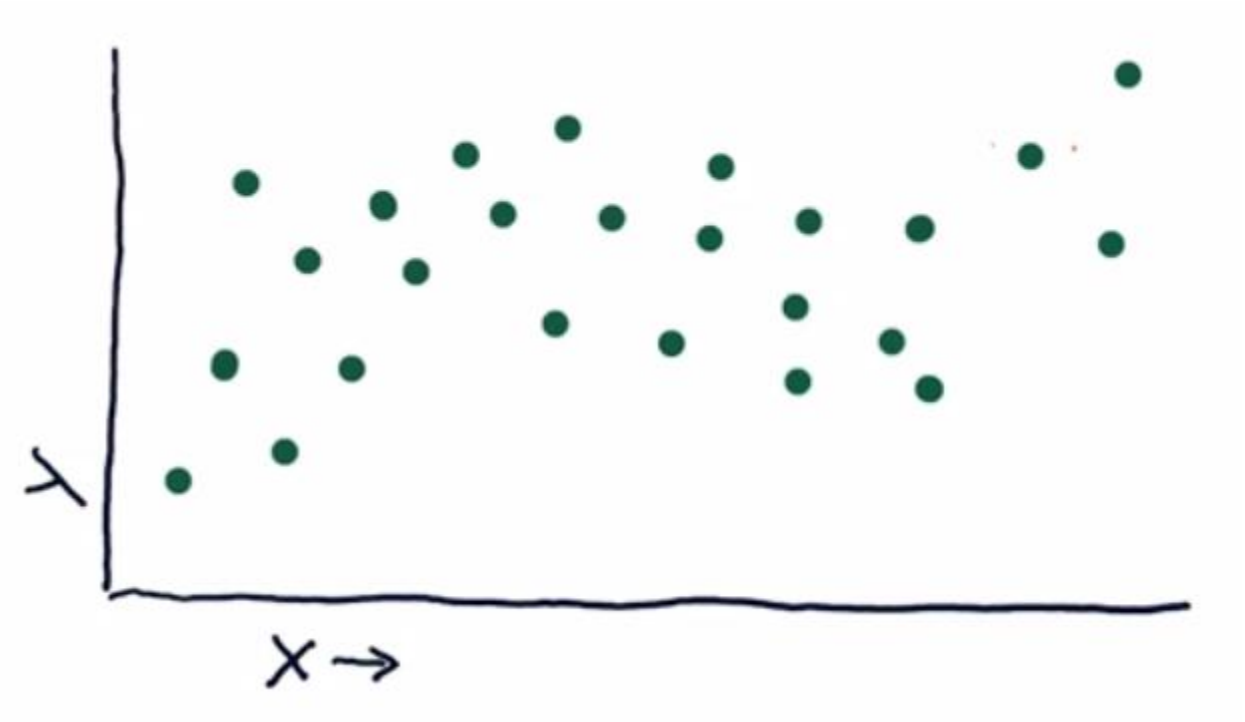
1. Build new random subset using bootstrapping (sampling with replacement)
2. Build a strong model on every bootstrap sample (high variance)

Final classification/regression is made using a vote/average method

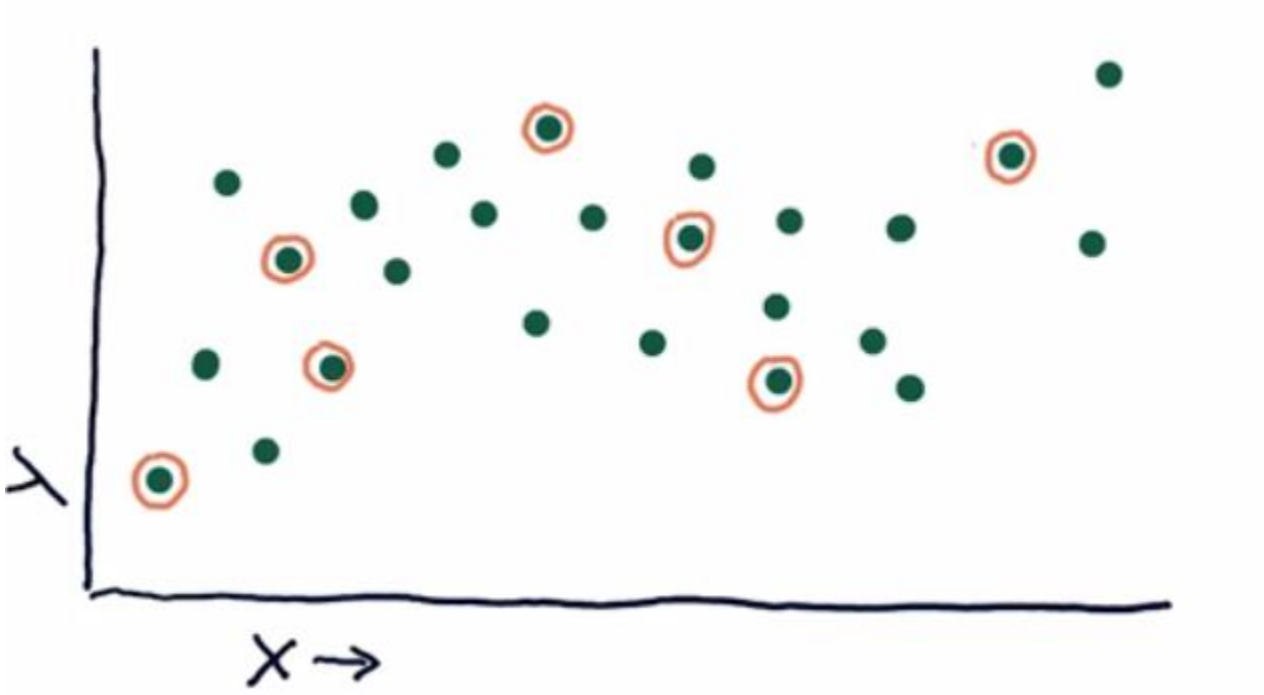
$$F_{final}(x) = \frac{1}{T}\sum_{t=1}^{T} f_t(x)$$



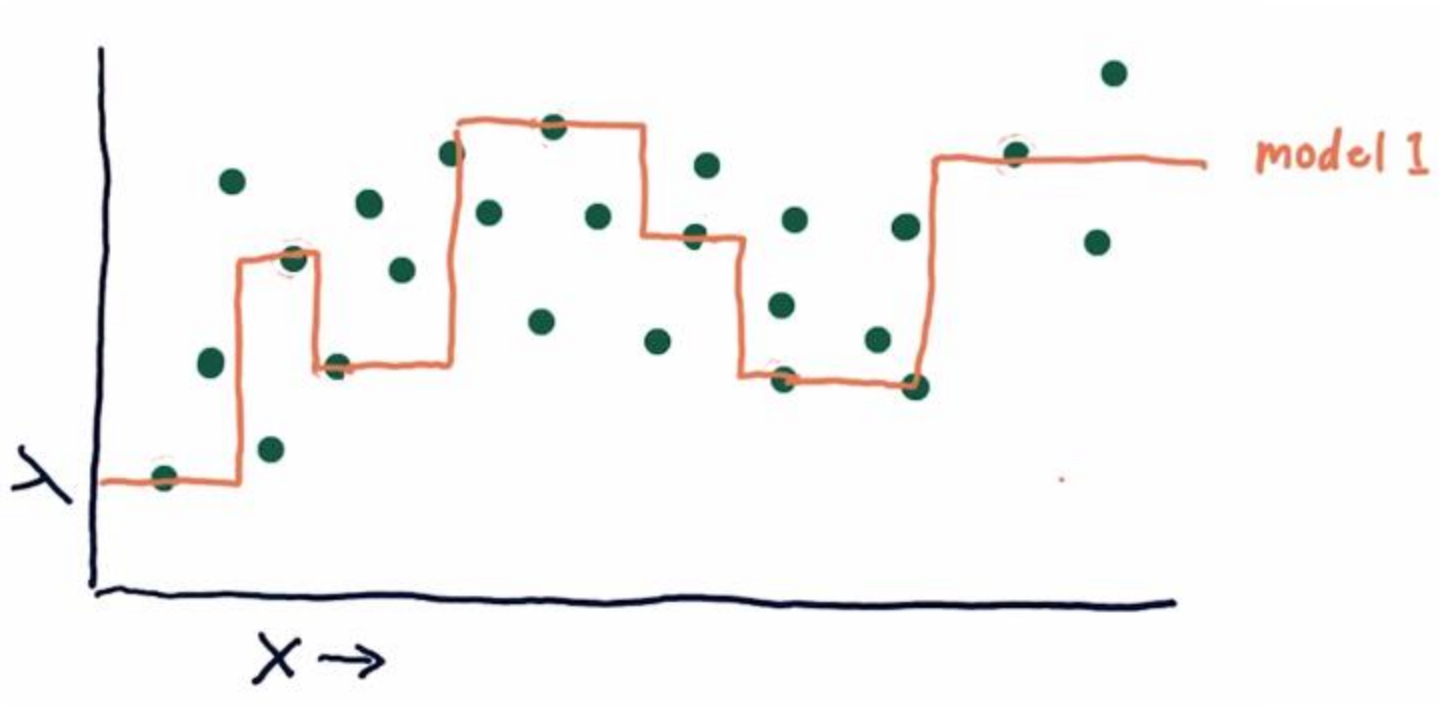
Bagging example



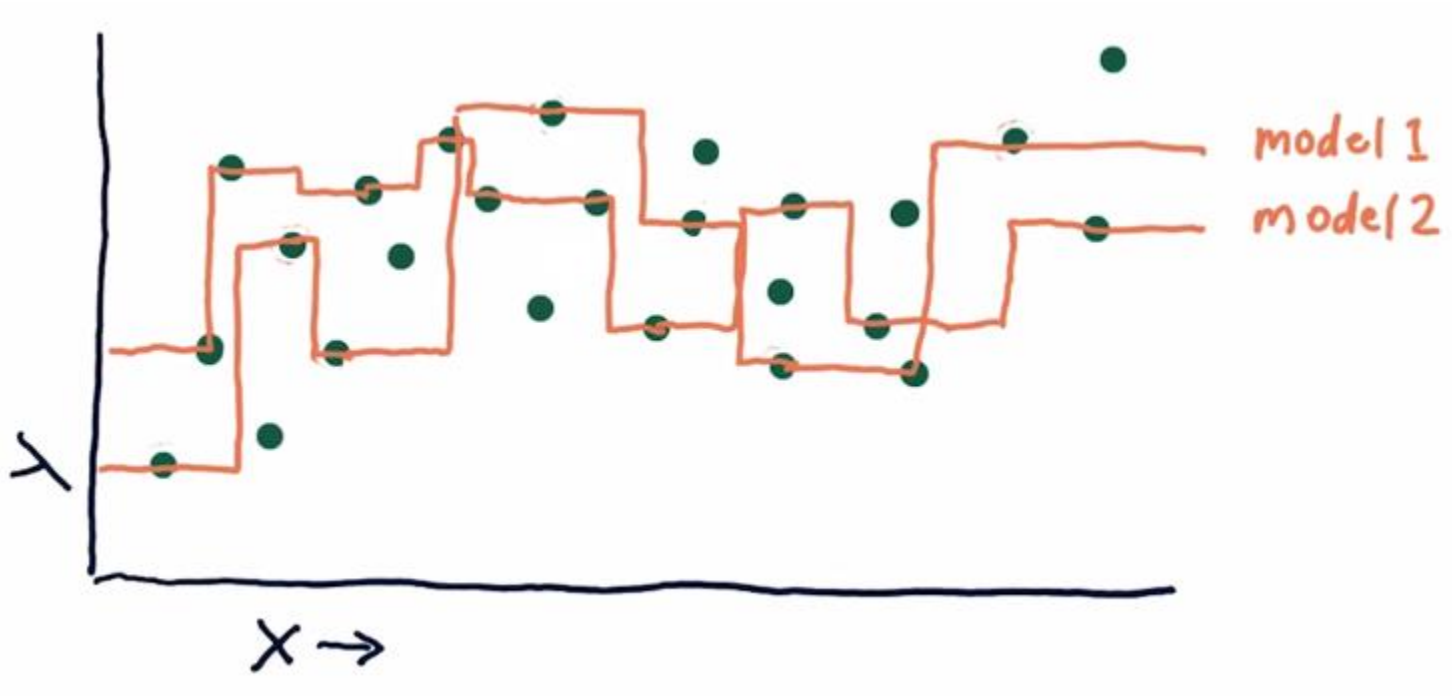
Bagging example



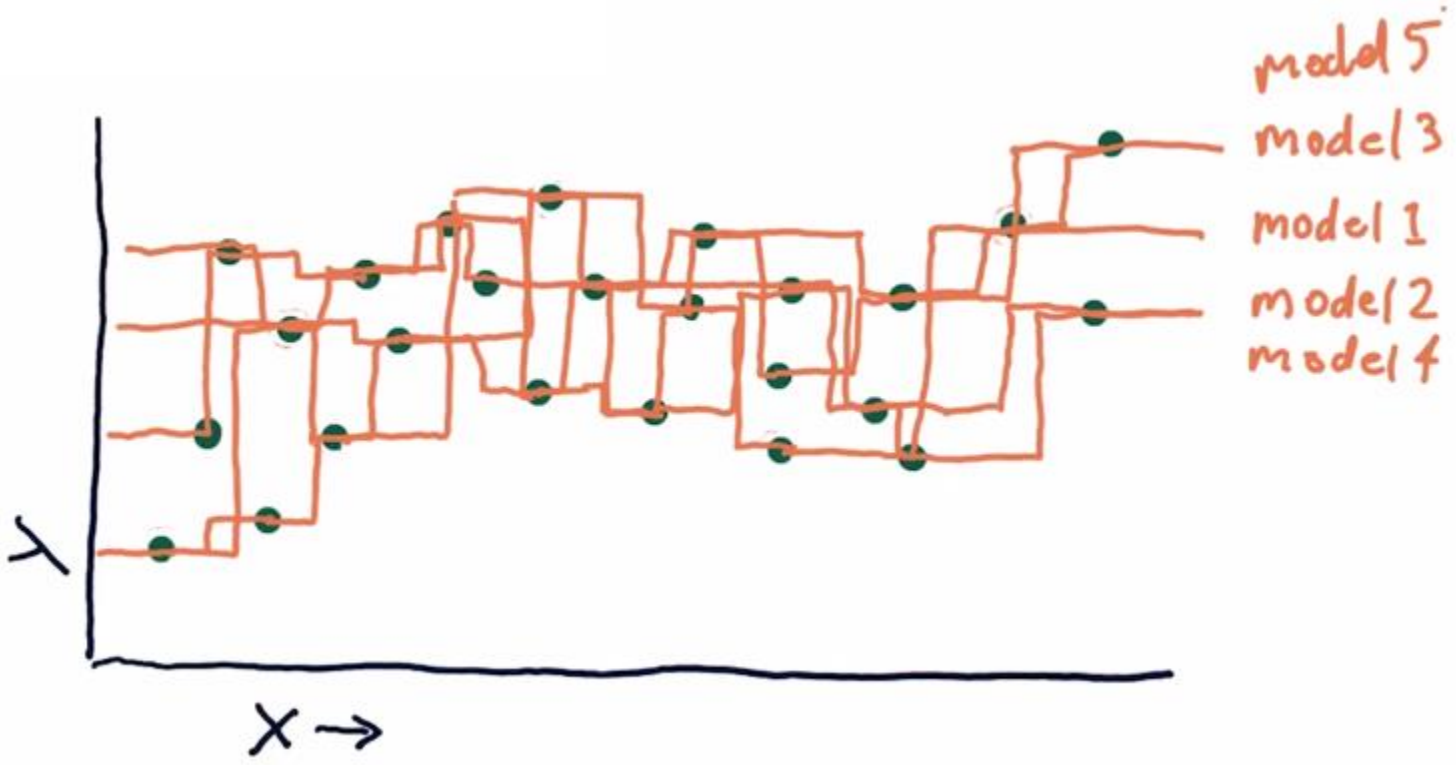
Bagging example



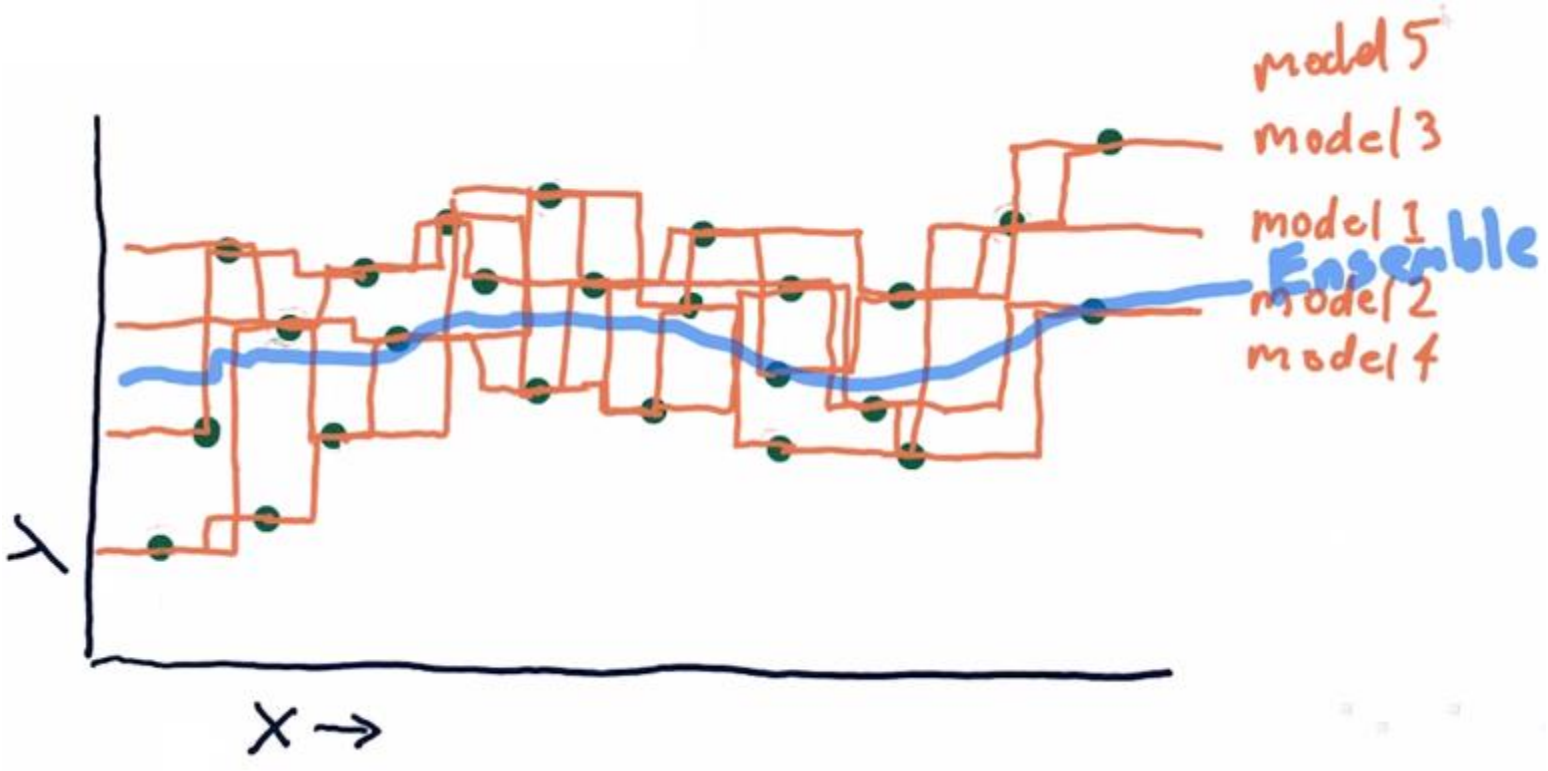
Bagging example



Bagging example

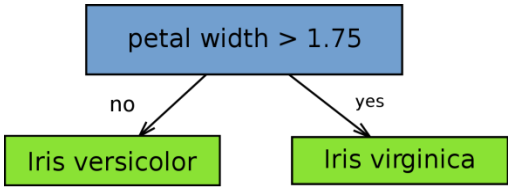


Bagging example



Ensemble methods

- Boosting :
- Handles the biais variance tradeoff by combining models with high biais and a low variance (e.g. decision stumps).



Decision stump

- Unlike bagging, the subsets are not randomly chosen but instead are chosen regarding the perfomances of the previous models.

- A famous algorithm is AdaBoost for adaptive boosting.

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$

Initialise weights $D_1(i) = 1/m$

For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

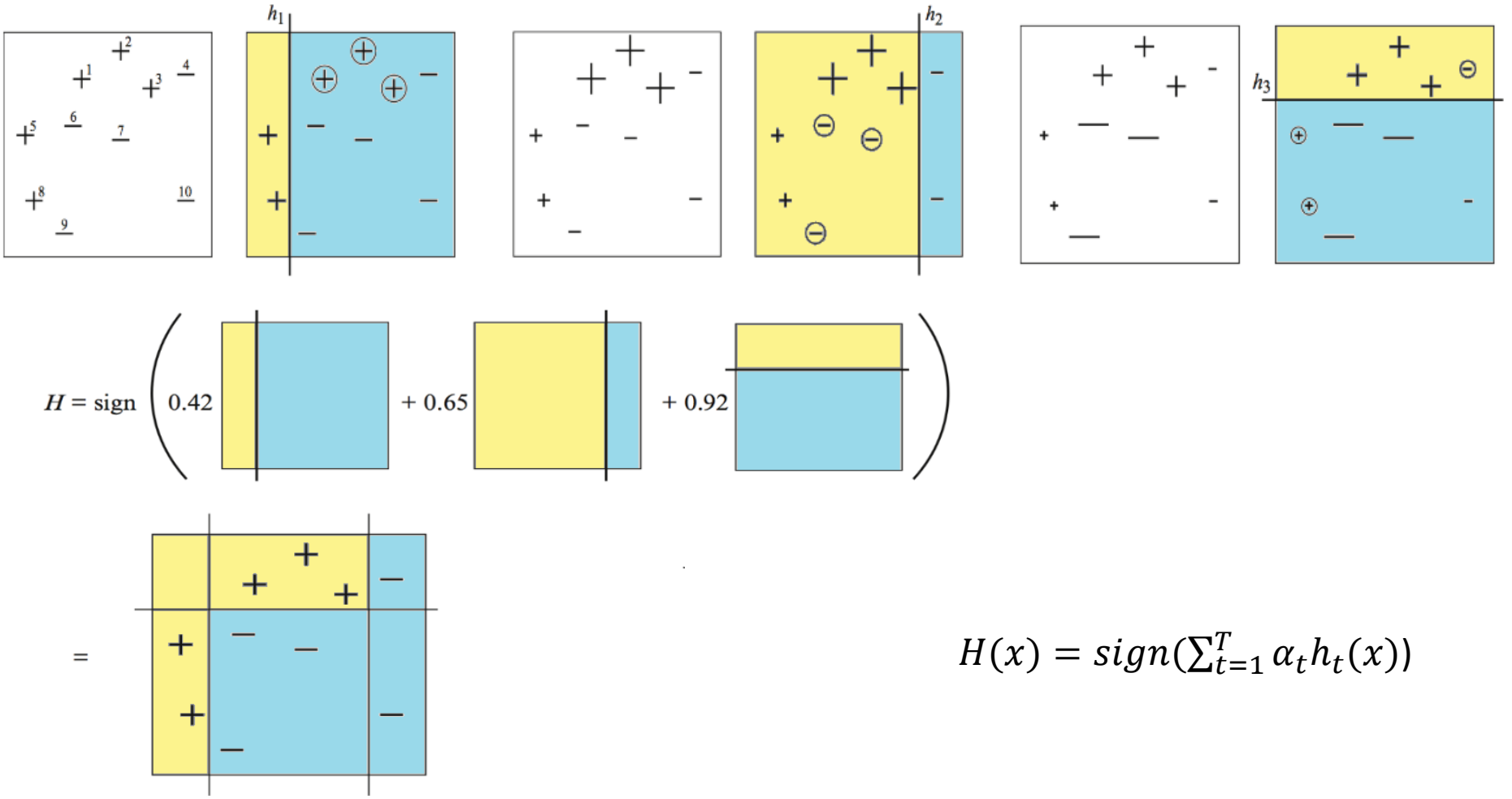
$$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Output the final classifier:

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

Adaboost algorithm

Boosting example



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

Gradient boosting

As adaboost, weak learners h are combined linearly:

$$f_t(x) = f_{t-1}(x) + \alpha_t h_t(x)$$

Adaboost is a special case of gradient boosting where the objective function is $\mathrm{R}_{\mathrm{emp}} = \sum_i e^{-y_i f(x_i)}$

At iteration $t-1$ we find the residuals for all $\{x_i\}_{i=1}^{M}$

$$g_t(x_i) = \frac{\partial L\big(y_i, f_{t-1}(x_i)\big)}{\partial f_{t-1}(x_i)}$$

Gradient boosting

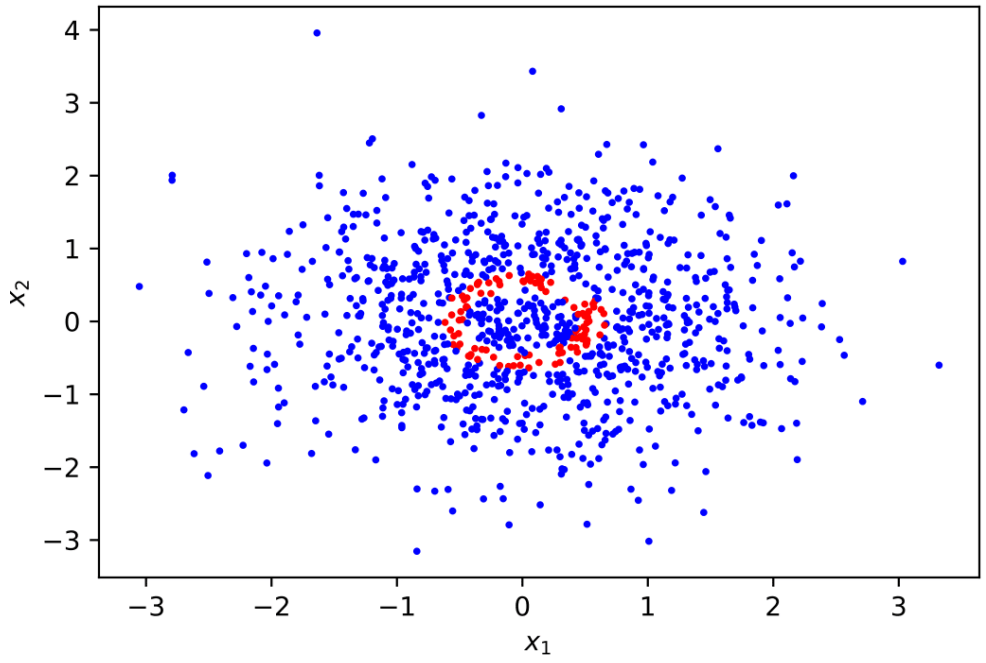
We find a model h_t with its corresponding weight such that:

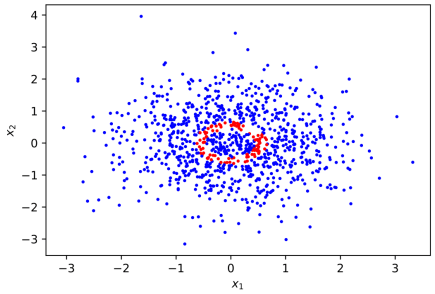
$$h_t = argmin_h \sum_{i=1}^{M} -g(x_i)h(x_i)$$

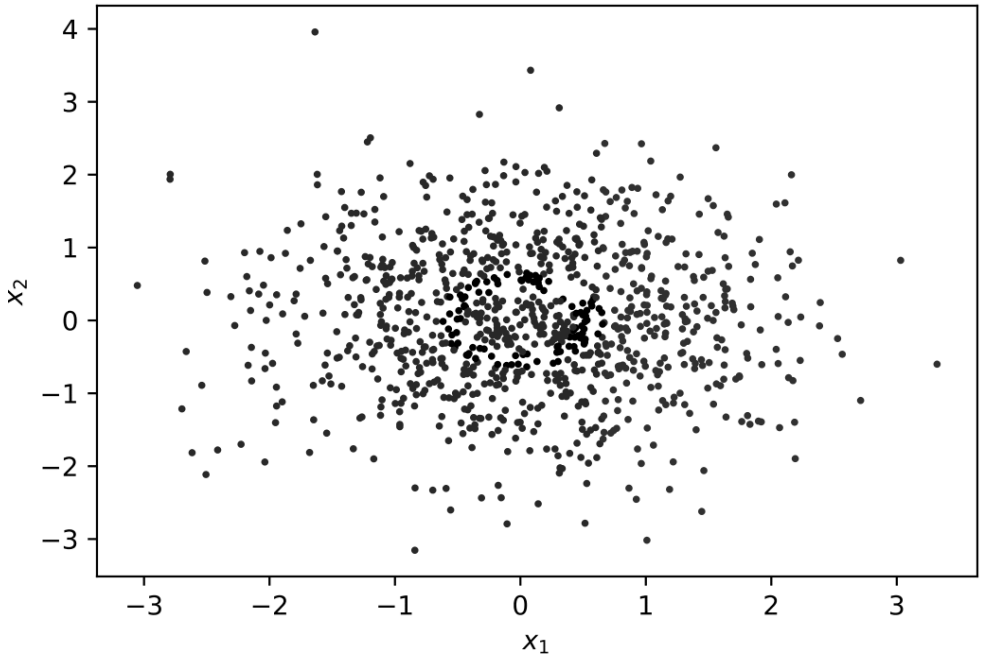
$$\alpha_t = argmin_\alpha \sum_{i=1}^{M} L(y_i, f_{t-1}(x_i) + \alpha h_t(x_i))$$

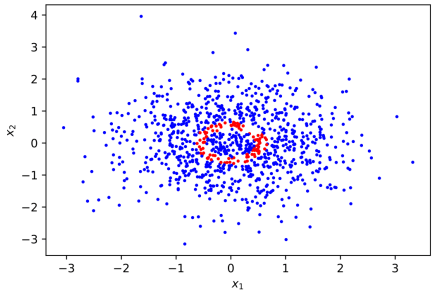
In our imbalanced context settings, the fact that new weak learners give more importance to the misclassified is a nice feature. At the end the minority class will get a lot of focused by the gradient boosting algorithm.

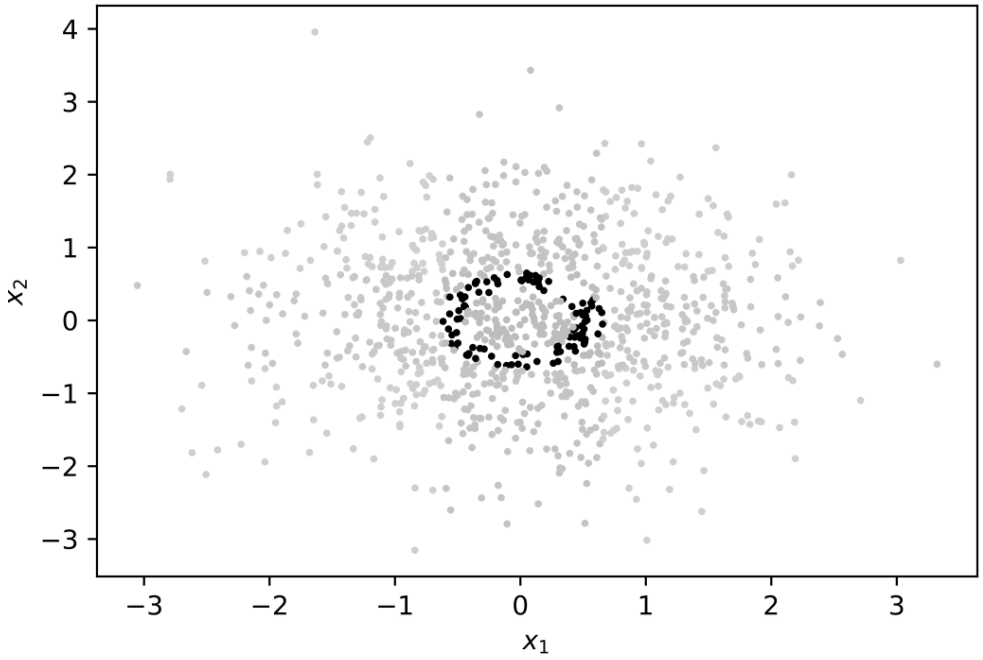
However, it does not entirely solve the problem of class imbalance.

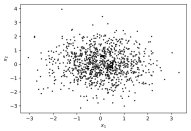

4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

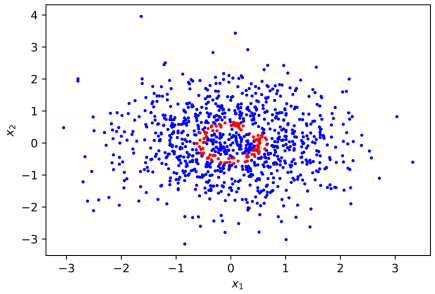

4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

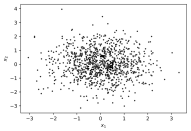

4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

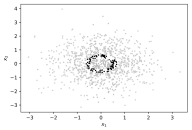

4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

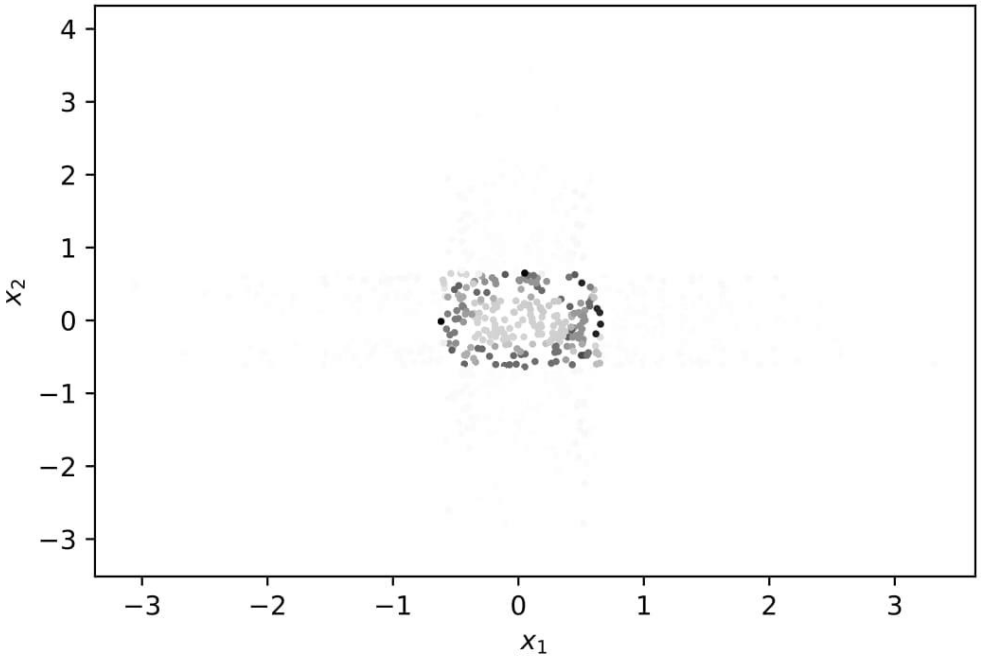

4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1



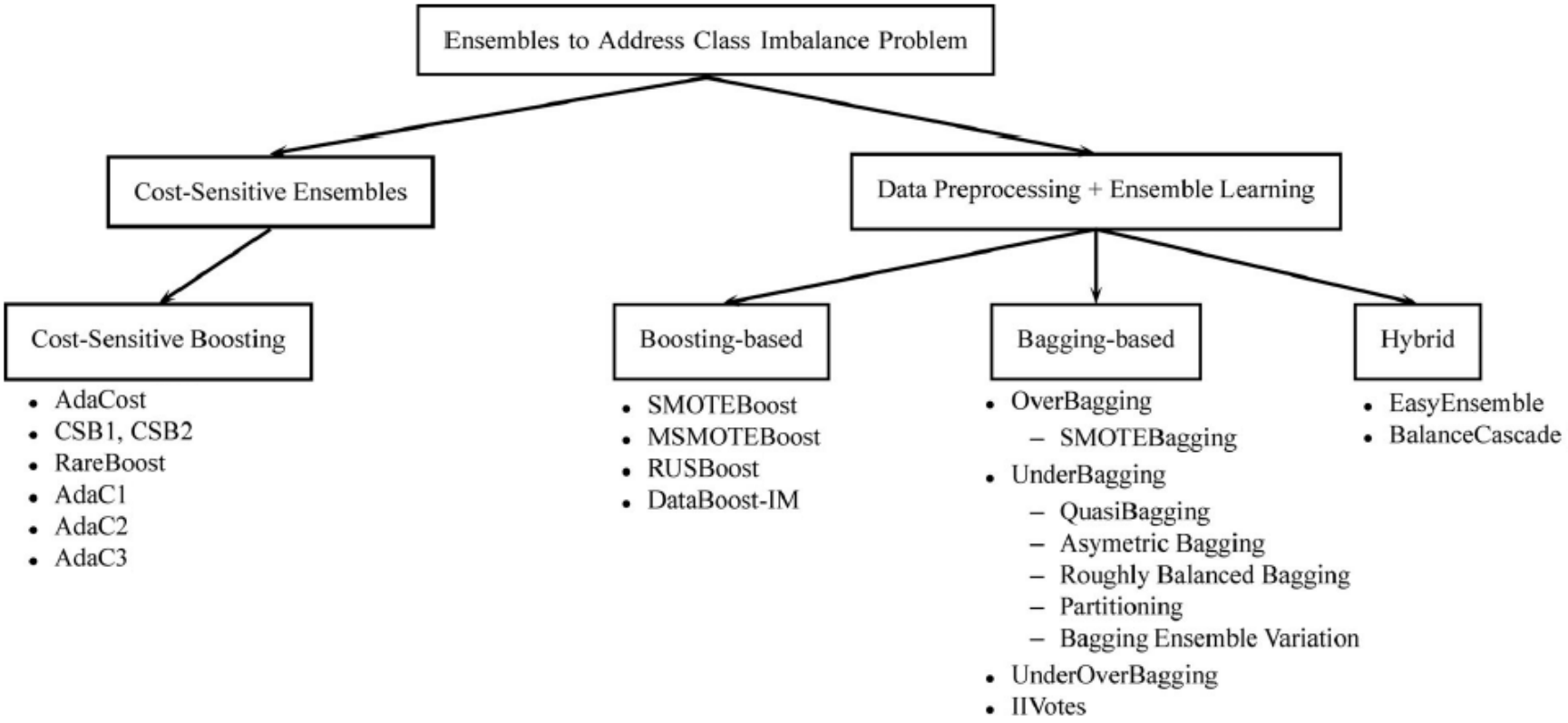

4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1






4
3
2
1
0
−1
−2
−3
x_2
−3
−2
−1
0
1
2
3
x_1

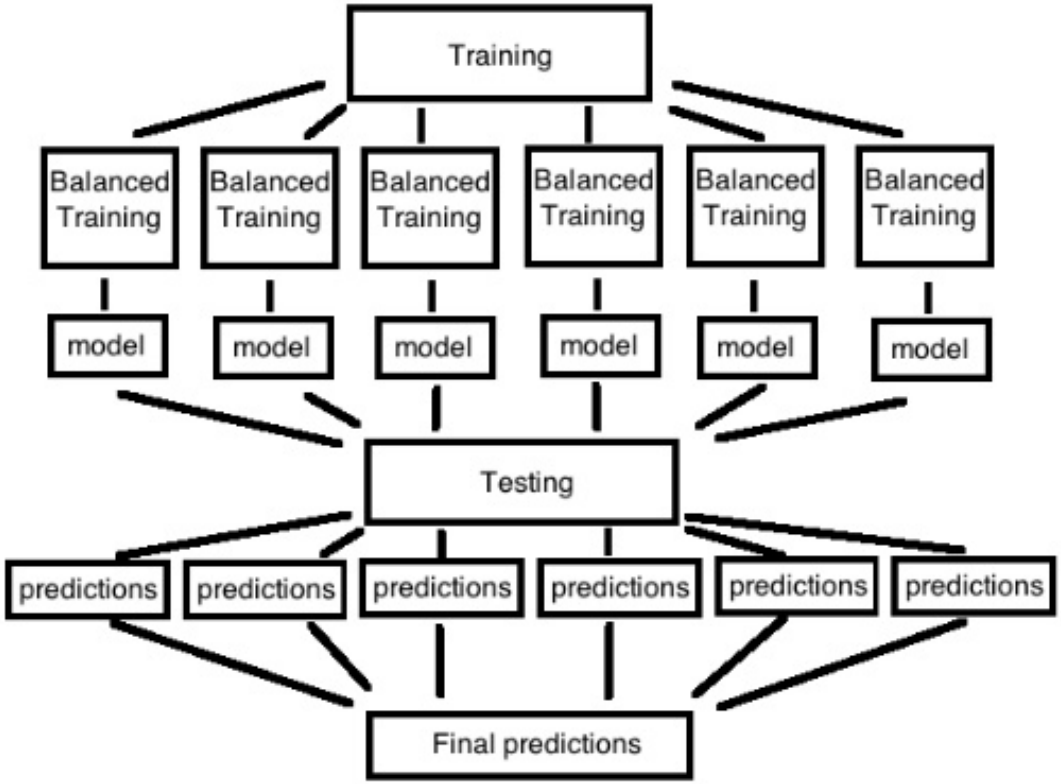
Ensemble methods for the class imbalance problem



Galar, Mikel, et al. "A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches." *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42.4 (2012): 463-484.

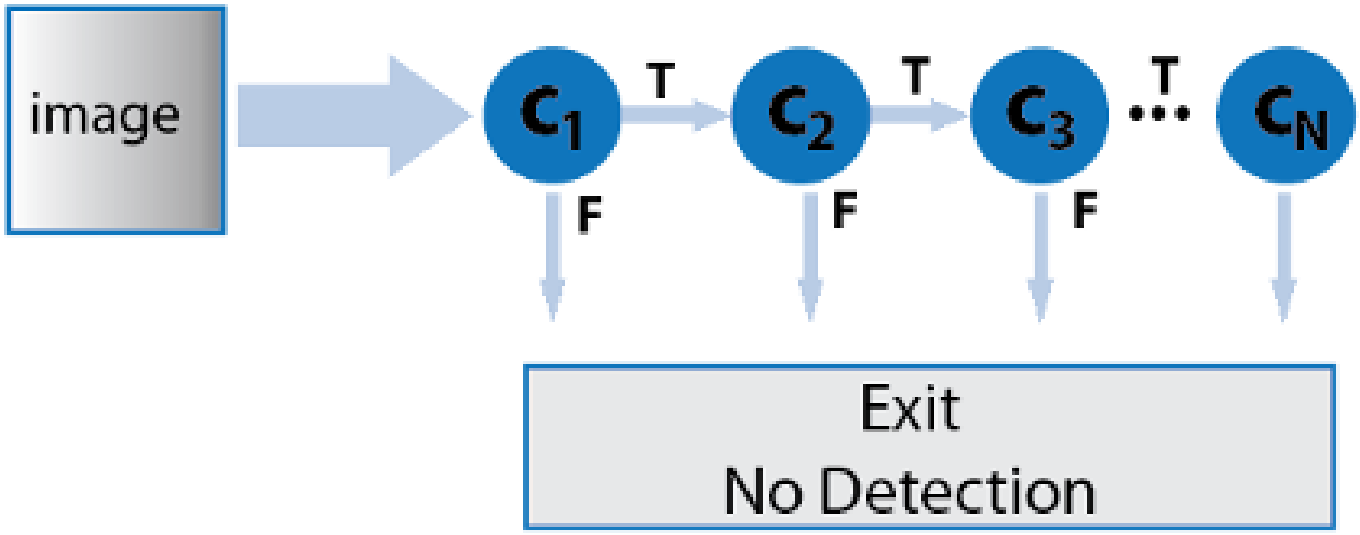
EasyEnsemble

- Simply creates several copies of different randomly balanced subsets



Cascade classifier

- Viola and jones detection algorithm



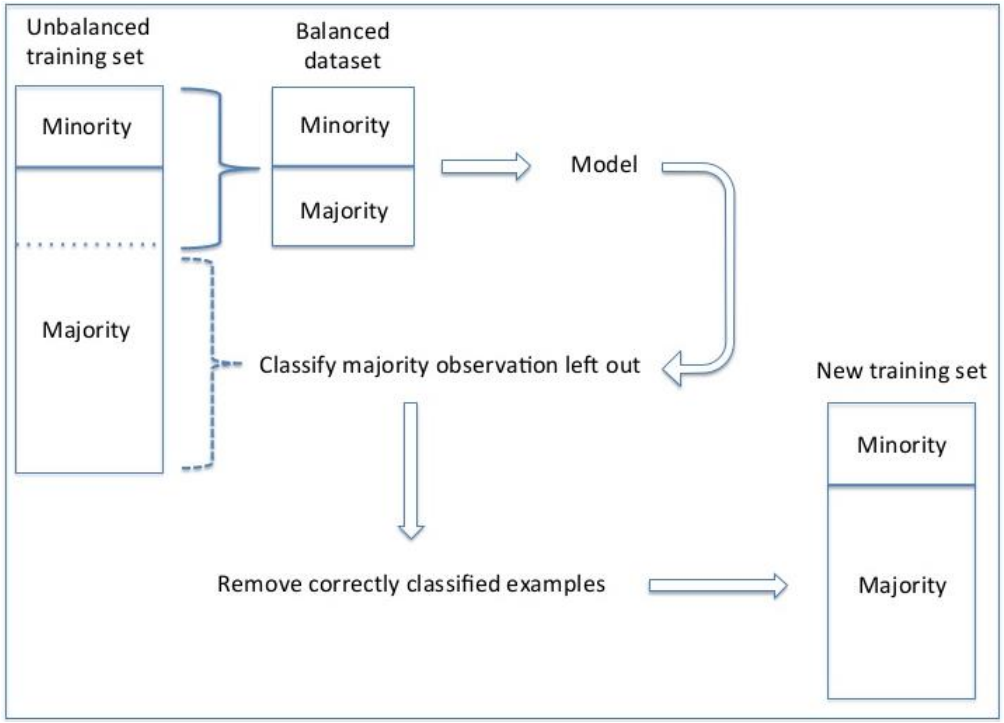
Balance cascade classifier

1. Generate $E \subset S_{maj}$ $(s.t. |E| = |S_{min}|)$, and $\mathrm{M} = \{E \cup S_{min}\}$, our new subset.

2. Learn C_t over $\{x_i, y_i\}_{i=1}^{M}$

3. Remove $\{x_i, y_i\}$ from S_{maj} where $C_t(x_i) = y_i$

4. Repeat 1. and learn C_{t+1} until stopping criterion is met



SMOTEBoost

- SMOTE + Boosting
- During the training of boosting the distribution D_t is modified by applying SMOTE.
- The process adds N examples for the new weak learner h_t. N is an hyper parameter.

Modify the distribution D_t with SMOTE

Given: $(x_1, y_1), \ldots, (x_m, y_m); x_i \in \mathcal{X}, y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
For $t = 1, ..., T$:

- Find $h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[\![y_i \neq h_j(x_i)]\!]$
- If $\epsilon_t \geq 1/2$ then stop
- Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
- Update

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

Output the final classifier:

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

Adaboost algorithm

Adacost

- Cost based + boosting
- When updating the distribution D_t to D_{t+1} we add a cost β such that :

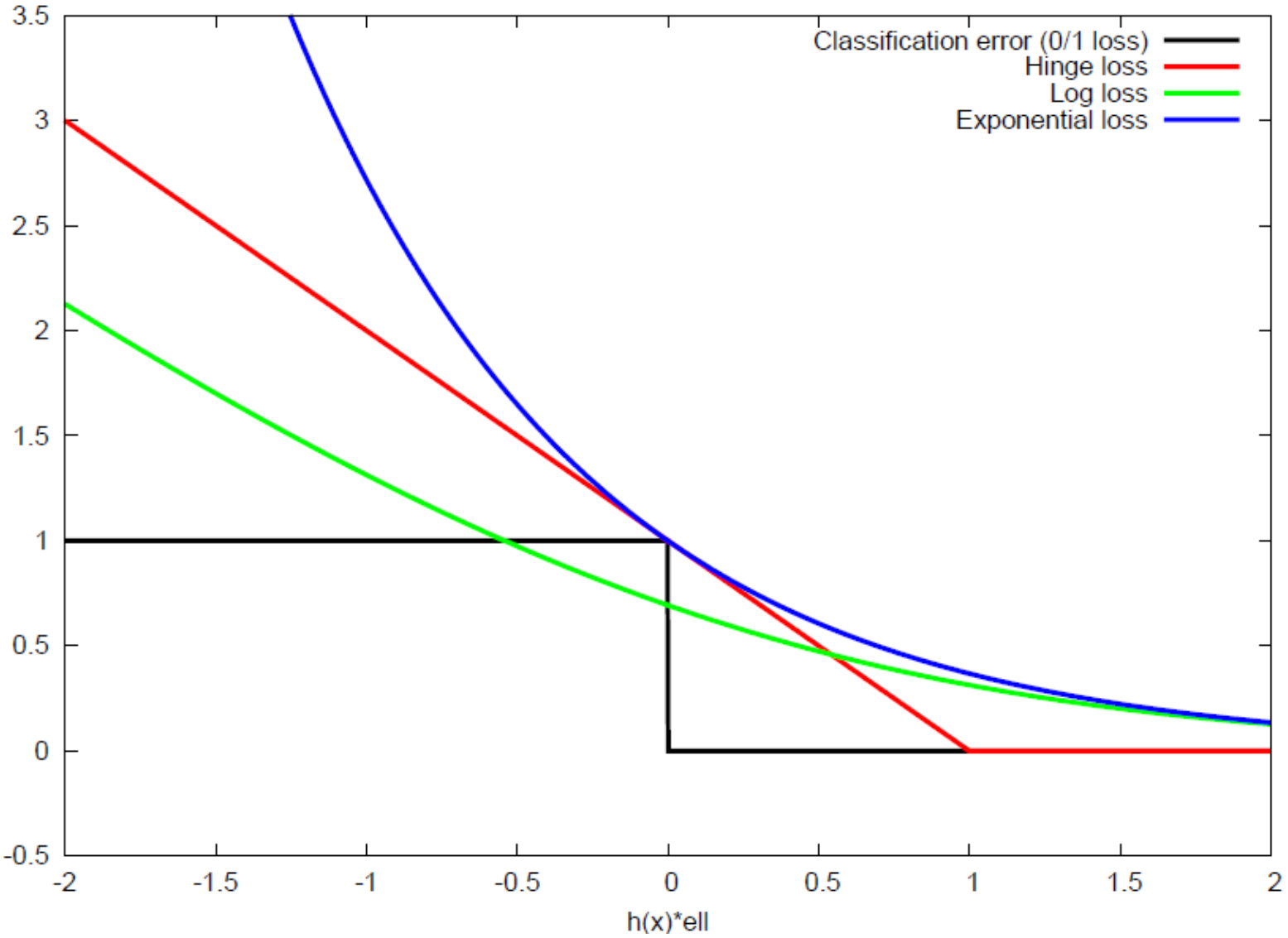
- $D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t h_t(x_i) y_i \beta_i)}{Z_t}$

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

Outline

- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Common losses



Example of different loss functions

Adapted objective function

- While difficult, creating new losses for the imbalance context is possible
- An interesting field for this: learning to rank

Learning to rank

- Used in information retrieval (e.g. web search)
- No more notion of classification
- Often based over queries
- The idea : optimize an objective function related to the rank.

RankNet

Optimise the ranking using neural networks

Notations:

- s_i, s_j the scores given by $f(x_i), f(x_j)$ respectively
- $P_{ij} = P(s_i > \mathrm{s_j}) = \frac{1}{1+e^{-\sigma(s_i - s_j)}}$
- y_{ij}, the known probability for $x_i$to be ranked higher than x_j

Ranknet optimises a special cost function:

- $C = \sum_{i,j} -y_{ij} \log P_{ij} \ - (1 - y_{ij}) \log(1 - P_{ij})$

LambdaRank/LambdaMART

- Pairwise measure tend to emphase more on relevent examples at the bottom of the ranking (black arrow)

- The idea of lambdaRank is to weight every pair by its absolute difference in the metric of choice when swapping s_i and s_j in the ranking

The objective function becomes:

- $C = \sum_{i,j} |\Delta Z_{ij}| (-y_{ij} \log P_{ij} - (1 - y_{ij}) \log(1 - P_{ij}))$

Learning to rank

Pros:

- Focus at ranking correctly the instances
- In most cases, optimises the top rank

Cons:

- No decision threshold -> no classification (you need to define one yourself)
- Might have a negative impact on the recall

Outline

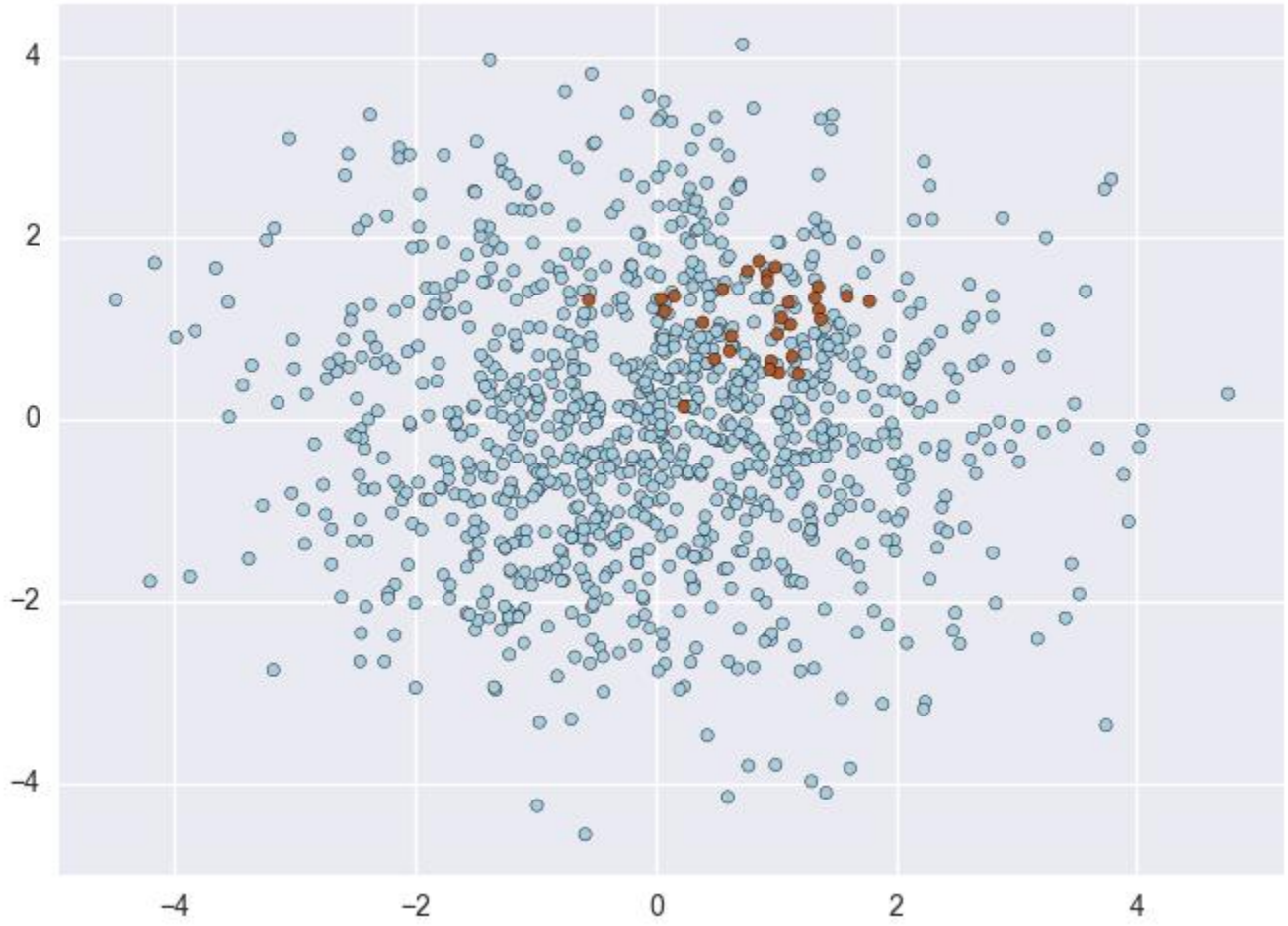
- Introduction
- Metrics for imbalanced learning
- Sampling methods
- Cost based learning
- Posterior probabilities
- Ensemble methods
- Adapted objective function
- Unsupervised learning
- TP

Unsupervised learning

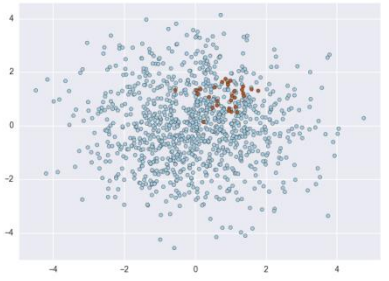
There exists solutions that are known as outlier detection:

- Classification based (e.g. One class SVM
- Clustering algorithms (e.g K-NN global anomaly detection)
- Statistical (e.g. histogram based, density based approach…)

Unsupervised ?



Unsupervised ?



- Unless the problem is now to detect *outliers*, it seems very hard to solve

- A solution could be to use metric learning/representation learning before hand

How to chose a strategy to work with imbalanced data ?

- There is no rules for handling imbalanced data since it depends on many factors :
  - Concept complexity
  - Degree of imbalance
  - Size of the datasets
  - The model

TP

- Download the dataset (credit card transactions).
- Choose two different strategies and build an output file on the test set in the following format:
  - predicted class, model score
- Send a the zip of your report comparing the strategies, your code and the output file to

jordan.frery@univ-st-etienne.fr

Dataset:

https://filezemp.univ-st-etienne.fr/snurrv41