Doc-Start
Doc-Start
Doc-Start
Doc-Start
Doc-Start
Doc-Start

Ensemble Methods in Machine Learning

Master 2 - MALIA

Guillaume Metzler

Université Lumière Lyon 2
Laboratoire ERIC, UR 3083, Lyon

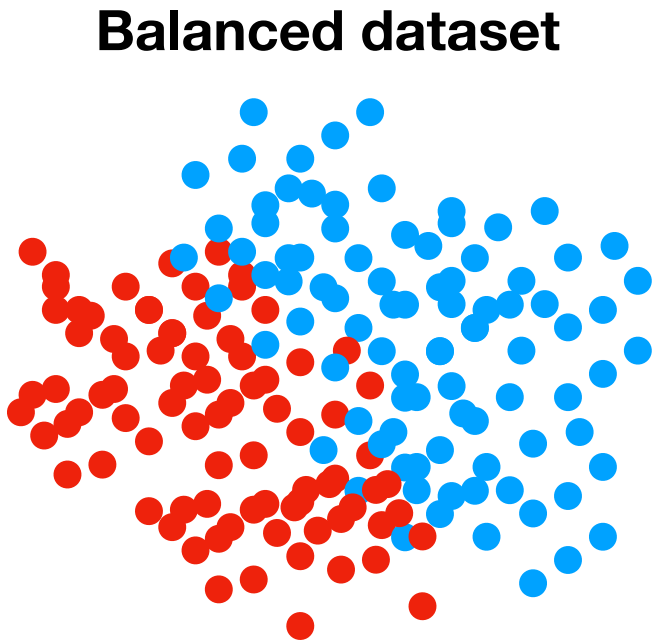
guillaume.metzler@univ-lyon2.fr

Fall 2023

Imbalanced Learning

Usual Context I

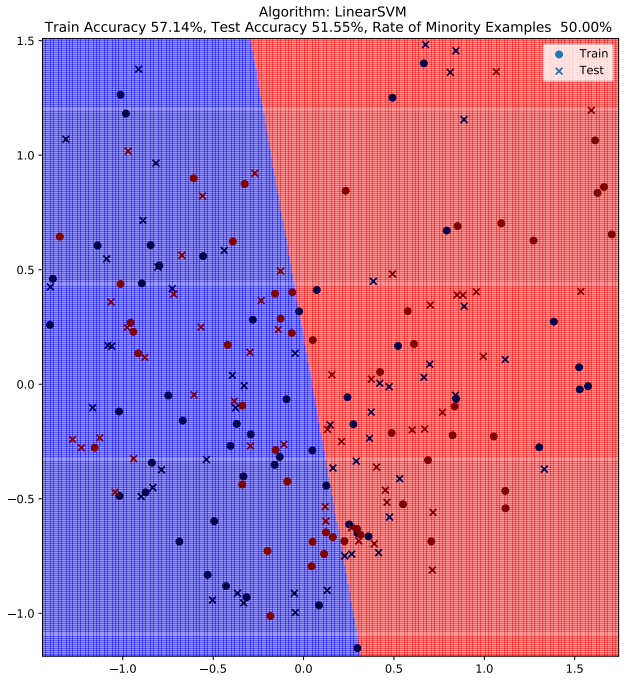
Most algorithms/methods used in Machine Learning assume that all classes are represented almost equally.

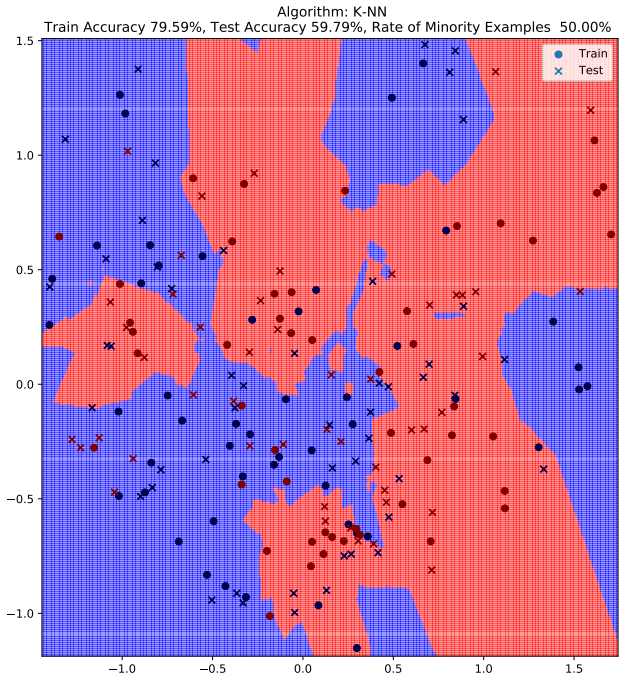


In general, the problem is said to be balanced and we have around 50% of each example in the two classes.

Usual Context II

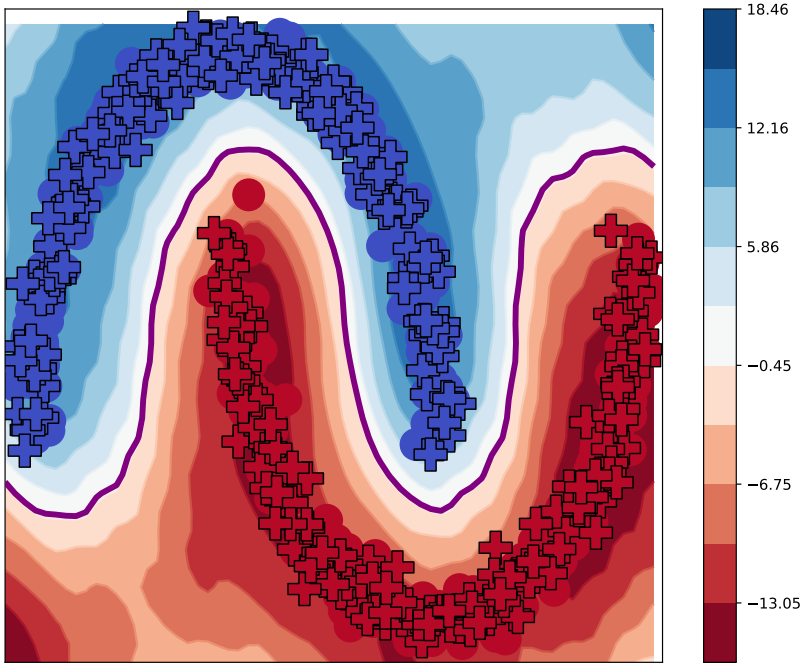
Standard algorithms, whether parametric or non-parametric, such as SVMs or k-nearest neighbors, provide reasonable performance on such data.





Usual Context III

Even more complex algorithms, such as non-linear SVMs with a Gaussian kernel, are therefore also able of solving our classification problems very well, even if the dataset is complex.



Usual Context IV

All these algorithms seek to minimize their error rate on the training set (except for the k-nearest neighbor).

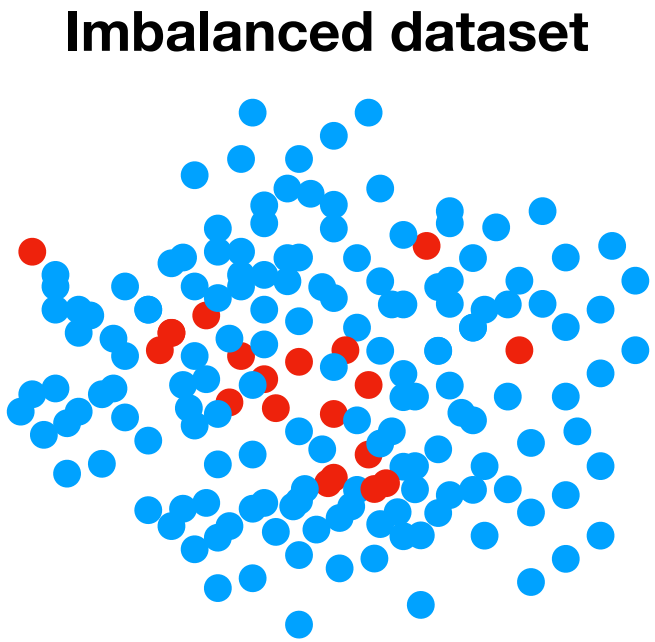
$$\mathcal{R}_S = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\{h(\mathbf{x}_i) \neq y_i\}}.$$

Or, more precisely, a surrogate of the *0-1* loss, *i.e.* upper bounds, via the use of loss functions

$$\mathcal{R}_S^{\ell} = \frac{1}{m} \sum_{i=1}^{m} \ell(h(\mathbf{x}_i), y_i).$$

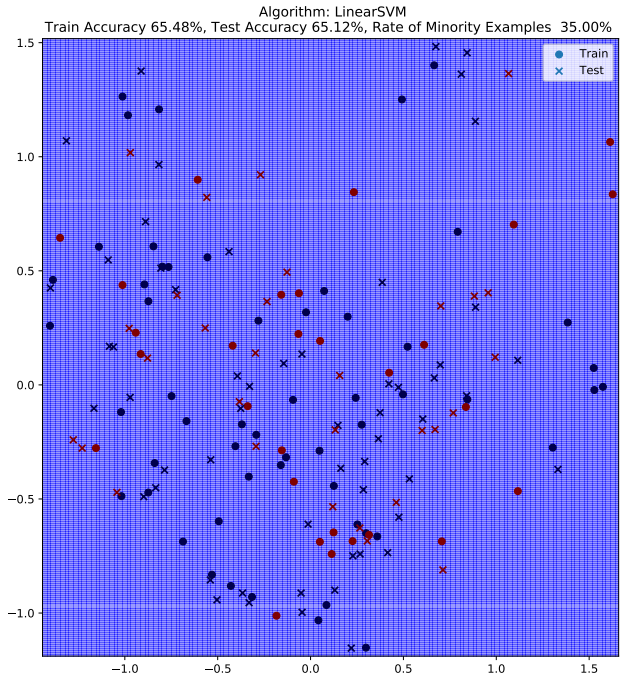
Minority class I

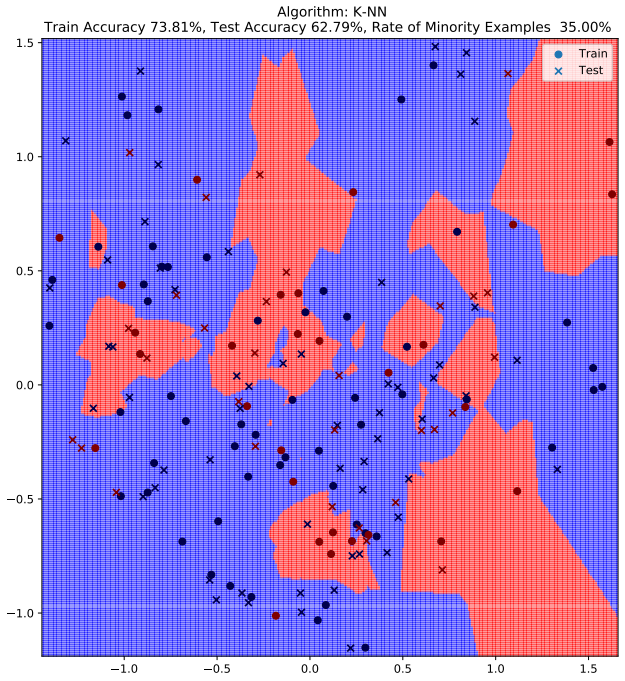
What happens when a class is under-represented compared to other classes ? When the number of examples in a class becomes very small compared to one or more other classes



Minority class II

How do our algorithms behave ? What can we say about the surfaces of our decisions, and above all... why ?





Minority class III

- Linear SVM tends to predict all data as negative, which has a direct impact on the algorithm's performance.
  The error rate is equal to the minority class representation. We have a performance close to 65%.
- The decision surface of a nearest neighbor tends to become unbalanced too, reducing the chances of finding examples of the minority class.
- Why ? The problem may lie with the algorithm or the criterion we're trying to optimize.

Minority class IV

In unbalanced problems, we're generally interested in finding examples of the minority class.
The latter may represent *frauds*, *anomalies* or *atypical examples* that could pose a threat.

Examples : bank fraud - tax fraud - intrusion into systems - anomaly on an assembly line - anomaly in medical analyses - anomaly in medical images - anomaly detection for measuring devices such as sensors, ...

Minority class V

By the way, can we say that frauds and anomalies are similar or not ?

Performance Measures

Performance Measure I

- Trying to minimize the error rate is not a good solution in this context, as it tends to favor the majority class.
- We prefer to evaluate the model with more suitable measures such as precision, recall or F-measure, using **our confusion matrix**

	$h(\mathbf{x}) = 1$	$h(\mathbf{x}) = -1$
$y = 1$	True Positive	False Negative
$y = -1$	False Positive	True Negative

Table – Confusion Matrix for a binary problem

Performance Measure II

- Precision : we evaluate the accuracy of the model's positive predictions

$$\frac{TP}{TP + FP}$$

- Recall : evaluates the model's ability to find examples of the positive or minority class

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

Performance Measure III

- F-measure : it's a harmonic mean between the two previous measurements, and depends on a parameter β which manages the trade-off between Recall and Accuracy (we often take $\beta = 1$).

$$\frac{1+\beta^2}{\frac{1}{Rappel}+\frac{\beta^2}{Précision}} = \frac{(1+\beta^2)TP}{(1+\beta^2)TP+\beta^2 FN+FP}.$$

where a *TP* is a fraud predicted as fraudulent by your model, a *FN* is a fraud not identified as such by your model, a *FP* is a non-fraudulent transaction but identified as fraudulent by your model and finally a *TN* is a non-fraudulent transaction.

Performance Measure IV

- You can also try to maximize the area under the ROC curve (AUC ROC). Its definition is slightly more complex : it's the help under the curve (TPR, FPR) where

$$TPR = \frac{TP}{TP + FN} \quad \text{et} \quad FPR = \frac{FP}{FP + TN}$$

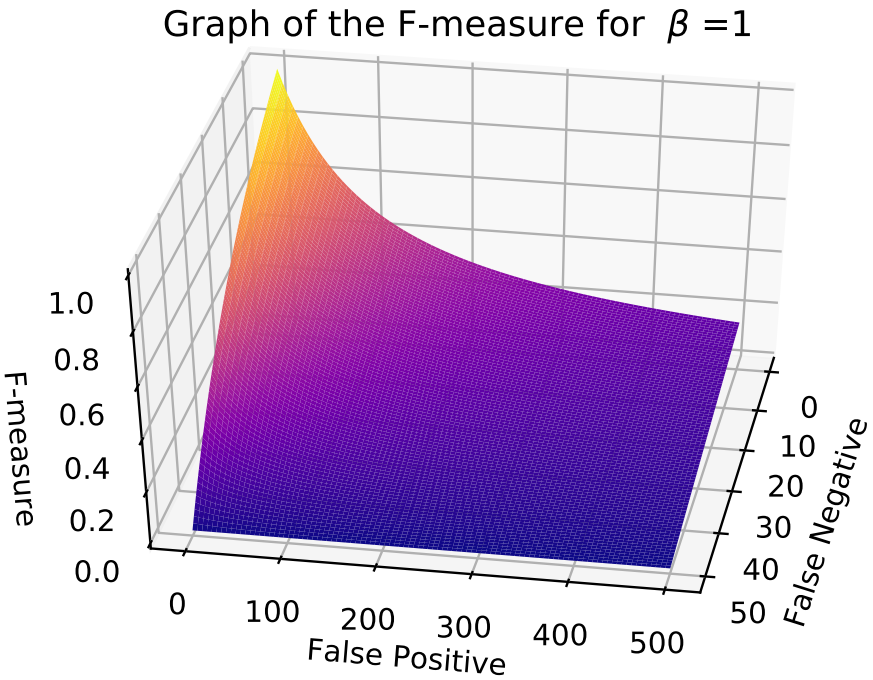
- Given a f model returning a score, we can also define the AUC ROC by the relationship :

$$\frac{1}{PN} \sum_{i=1}^{P} \sum_{j=1}^{N} \mathbb{1}_{\{f(\mathbf{x}_i^+) \geq f(\mathbf{x}_i^-)\}}.$$

The latter provides more information on the overall quality of the model, and is also better suited to ranking issues.

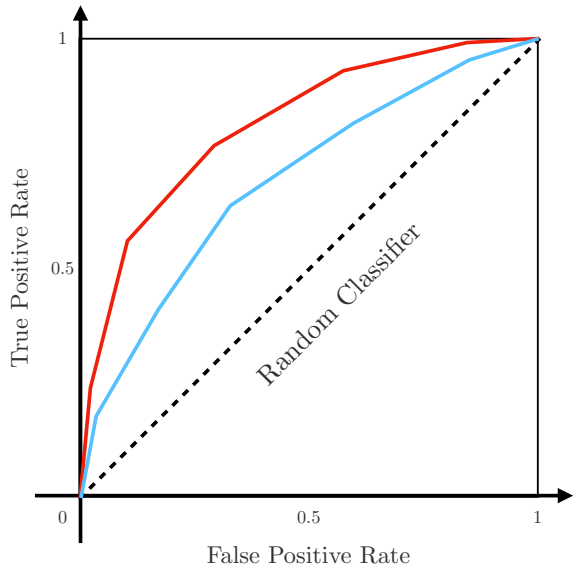
Performance Measure V

F-Mesure



$$F_\beta = \frac{(1+\beta^2)(P-FN)}{(1+\beta^2)P - FN + FP}$$

AUC



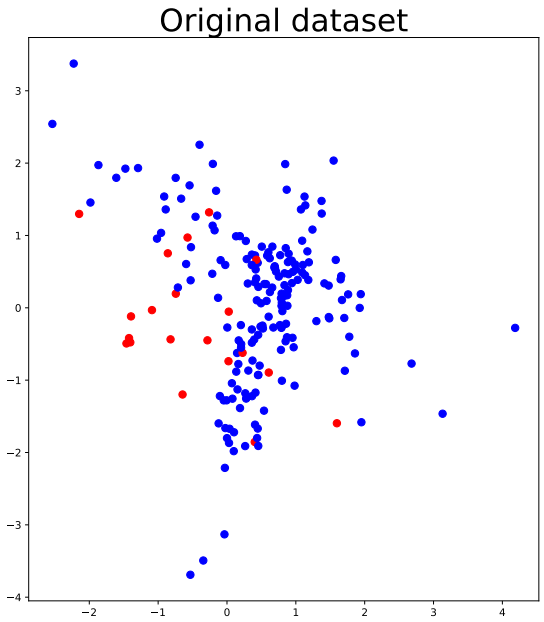
$$\mathbb{P}[f(x_+) > f(x_-)]$$

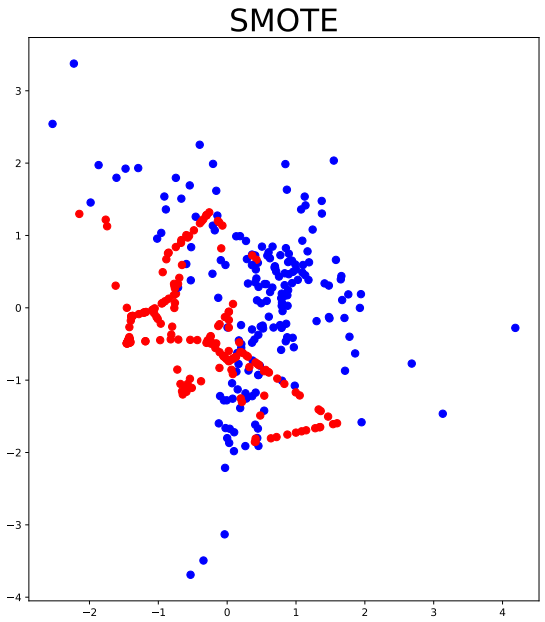
The latter provides more information on model quality, and is also better suited to ranking issues.

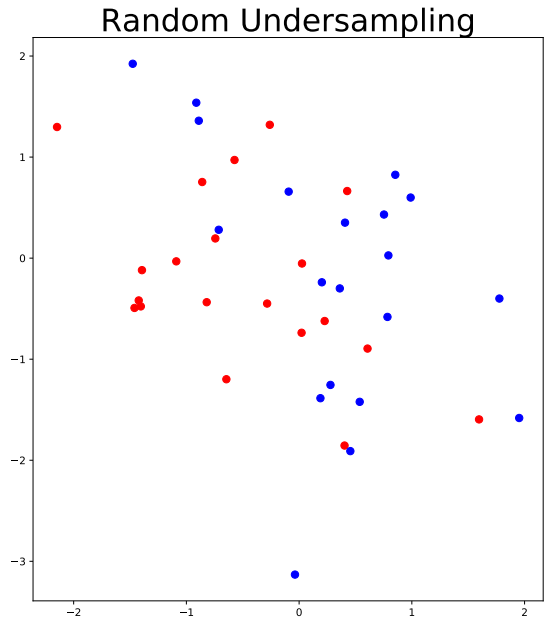
Sampling Strategies

Learning in an Imbalanced setting

Use sampling strategies







- Oversampling : Random - SMOTE - BorderSMOTE, ...
- Undersampling : Random - Tomek Link - ENN, ...

Oversampling I

Random

This is certainly the simplest approach to use. It doesn't involve randomly generating points, but simply duplicating existing points of the minority class.

So we're going to ask our algorithm to focus more on the areas of space where these points have been replicated.

Unfortunately, in doing so, we don't add any information to our data and we don't really control what happens... the process is totally random. Let's see how we can add information.

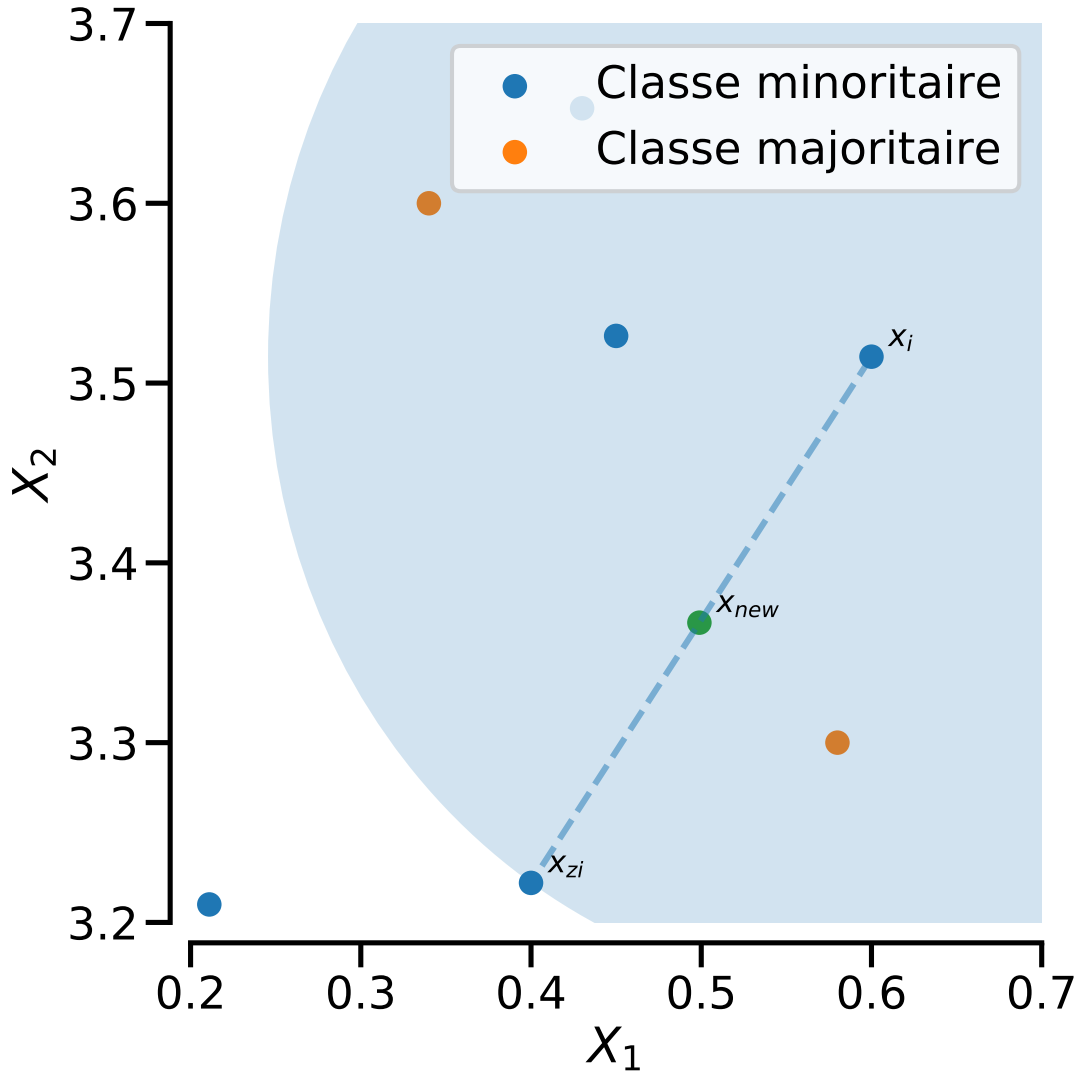
Oversampling II

SMOTE

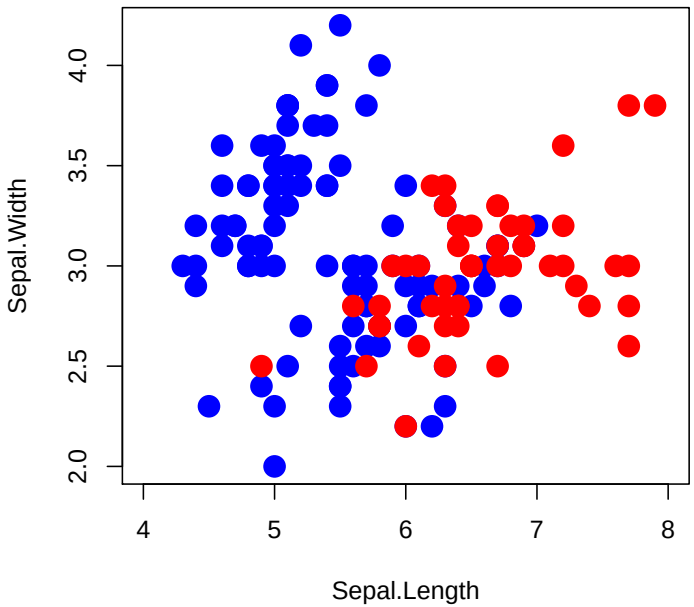
Algorithm 1: SMOTE Algorithm [Chawla et al., 2002]

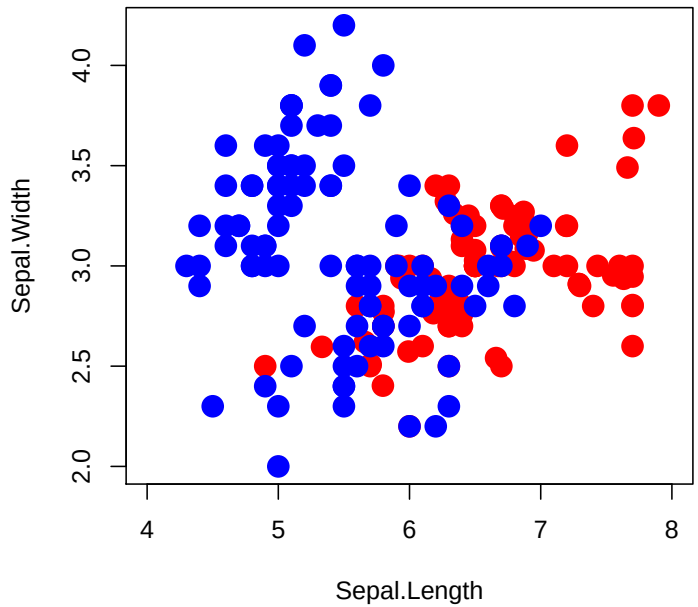
Input: Learnng sample S of size $m = p + n$,
k : number of neighbors, R rate of positives to generate
Output: A sample S'
begin
  Set $New = R \times p$: the number of positives to generate Syn,
  select randomly New examples among p et denote by $setind$ their indexes
  for $i \in setind$ **do**
    find the k-nn positives $\mathbf{p}_i$,
    randomly select one of the nearest neighbor rNN_i **for** *attributes* $attr$ *de* $\mathbf{p}_i$ **do**
      choose $\alpha \in [0, 1]$,
      set $newattr = \alpha p_i[attr] + (1 - \alpha) rNN_i[attr]$
      set $Syn_i[attr] = newattr$
    Then set $S' = S \cup Syn_i$
  Set $S' = S$
return S'

Oversampling III



Oversampling IV





Oversampling V

BorderlineSMOTE

The idea is very similar to SMOTE, but this time we'll impose a selection rule on the examples of the minority class that will be used for data generation.

- We set two parameters corresponding to a number of neighbors k and k'. The first is specific to SMOTE, and the second is used to determine the **borderline** examples.
- A borderline example is one in which the majority of neighbors (but not all !) belong to the majority class.
- For these examples, we'll then generate points using the SMOTE procedure.

Oversampling VI

In algorithmic terms, we will therefore :

1. detect all examples in the minority class that have at least $k'/2$ neighbors in the majority class (but not exactly $k'/2$)
2. for all these examples we'll apply the SMOTE algorithm to achieve the desired ratio of positives.

This algorithm will therefore focus on examples in the minority class that are difficult to classify, and leave out examples that are easy to classify (majority of neighbors in the minority class) or that are noise (k' neighbors in the majority class).

Oversampling VII

ADASYN

This is a smoother version of Borderline SMOTE. The idea is very similar, but this time the aim is to concentrate the generation on examples that are more difficult to classify.

We will define a risk score r_i on the examples of the minority class which corresponds to the number Δ_i of neighbors in the majority class out of the total number of neighbors, *i.e.*

$$r_i = \frac{\Delta_i}{k'}$$

The higher the score, the more examples will be generated around this data.

Oversampling VIII

If we want to create a total of N examples, we'll distribute them over all the data in the minority class. So we need to transform our r_i risk scores into a distribution.

$$r'_i = \frac{r_i}{\sum_{j=1}^{p} r_i}$$

And so, for each example of the minority class $\mathbf{x}_i$ we'll create exactly $N \times r'_i$ examples.

Oversampling IX

Implementation

All the above methods can be found in the library *imblearn.over_sampling* and the associated functions are :

- **RandomOverSampler**
- **SMOTE**
- **BorderlineSMOTE**
- **ADASYN**

For all these functions, the number of examples generated is determined by the final ratio of positives desired in our dataset

$$\frac{n_{\text{positives after sampling}}}{n}.$$

Oversampling X

to finish with [Fernández et al., 2018]

- Generates new examples of the minority class.
- For some variants, you can even concentrate on difficult areas (the decision boundary).
- But this increases the number of examples
- It also sometimes generates noise in the data

Undersampling I

Aléatoire

We will randomly delete or remove examples from the minority class in order to reduce the imbalance, but without affecting the majority class this time.

This is likely to make the decision boundaries between the minority and majority class examples much clearer.

Unfortunately, in doing so, we run the risk of losing information, and sometimes a lot of it, as the process is totally random.
Let's see how we can set up a good rule for deleting these examples.

Undersampling II

Condensed Nearest Neighbor [Hart, 1968] :

This is a pre-processing method used to speed up the nearest neighbor algorithm.

Algorithm 2: Condensed Nearest Neighbor

Input: Learning sample S
Output: A sample S' smaller than S
begin
  Seperate s into to sets S_1 and S_2
  while S_1 *anc* S_2 *are modified* **do**
    remove from S_1 tall the missclassified examples using S_2 and a 1-NN
    remove from S_2 tall the missclassified examples using S_1 and a 1-NN
  $S' = S_1 \cup S_2$;
  return S'

Undersampling III

Algorithme Tomek Link [Tomek, 1976]

This is a data cleansing method similar to CNN. To do this, we consider

pairs of examples with different labels. If the two examples are closer neighbors of each other, we say they form a *link Tomek*.
space0.3cm. These examples are therefore at a border and can potentially

lead to misclassification. We therefore decide to remove them from our

dataset.
We can also choose to remove only examples from the majority class !

Undersampling IV

Edited Nearest Neighbor [Wilson, 1972]

This is a very similar version of Tomek Link. The idea is to set a parameter k which corresponds to the number of neighbors to be considered for the cleaning process (generally $k = 3$).
For each example of the majority class, we'll :

- determine its k- nearest neighbors
- determine the majority class in the k-neighborhood
- delete the example from the training set if the prediction does not coincide with the true label

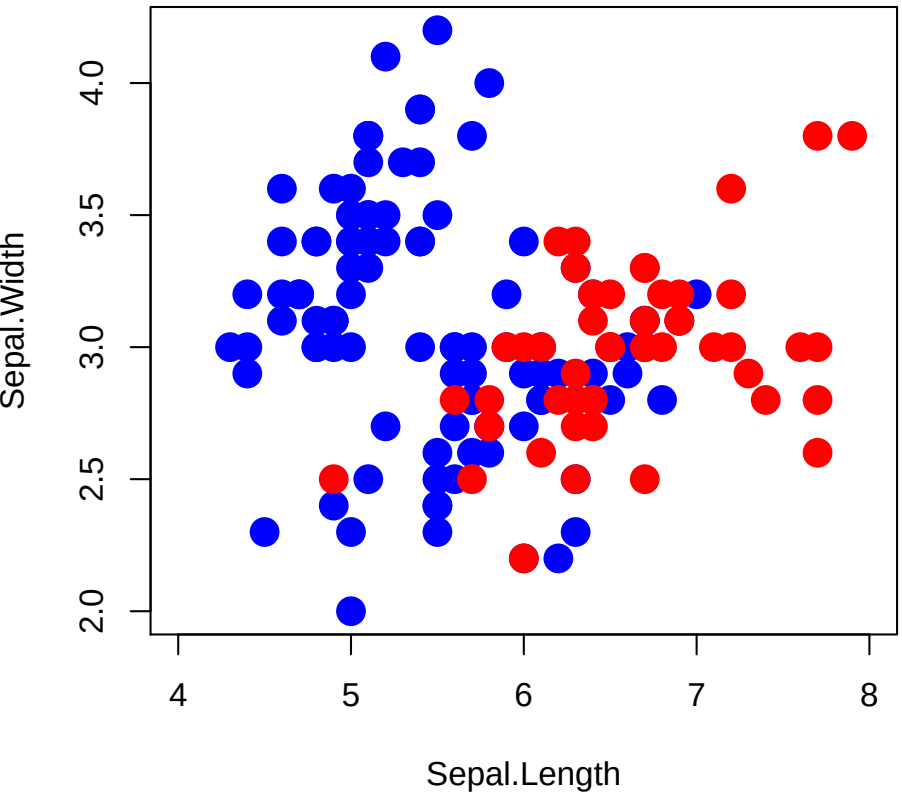
Undersampling V

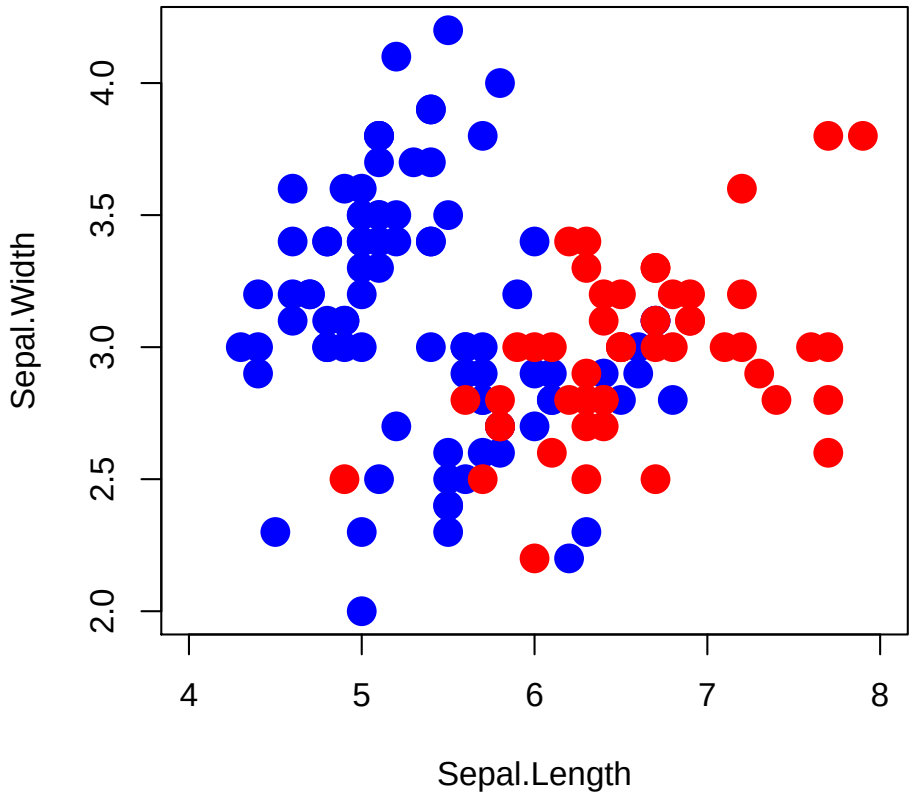
One Sided Selection [Tomek, 1976]

This last approach consists in applying the *Condensed Nearest Neighbor* algorithm followed by *Tomek Link* to reduce the computation time of the Tomek link approach when the datasets are large.

It also reduces the size of the dataset, which can be very interesting from an algorithmic point of view, enabling us to learn our models more quickly.

Undersampling VI





Cost-Sensitive Learning

Method I

This second approach consists in modifying the way errors are taken into account by our algorithm elkan2001foundations.

We'll assume that errors don't have the same weight.

If we want our algorithm to focus on examples from the minority class, we'll have to tell it that misclassifying such an example will result in a significant loss value, and that, on the contrary, misclassifying an example from the majority class will have a lesser consequence.

Method II

The weighting is done by first defining a cost matrix as shown below. It's similar to a confusion matrix, but instead of looking at the number of good/bad decisions made by the algorithm, we'll associate weights/costs with the decisions it makes.

	$h(\mathbf{x}) = 1$	$h(\mathbf{x}) = -1$
$y = 1$	c_{TP}	c_{FN}
$y = -1$	c_{FP}	c_{TN}

Method III

In our current context, we'll be focusing on the values of c_{FN} and c_{FP}, which define the weight of errors made on the minority and majority class respectively.

For example, if we set $c_{FN} = 2$ and $c_{FP} = 1$, then misclassifying an example from the minority class will cost twice as much as misclassifying an example from the majority class. By doing so, we'll be able to give more importance to the minority class.

Method IV

Implementation

In standard implementations, we always have $c_{FN} = c_{FP} = 1$, *i.e.*, all errors have the same weight.

There is no specific implementation for cost-sensitive learning, so you'll usually have to set the '*class_weight*' parameter in the various Python implementations.

Method V

Remarks

We could go further in error weighting and be more refined. If we assign different weights to each class, there's nothing to stop us from defining different weights for each example regardless of its class.

In an economic or even banking context (insurance, tax or banking fraud), it's not uncommon to assign a higher weight to high-dollar frauds than to small ones ! That's what you do if you want to minimize losses during the fraud detection process.

Optimizing F-measure

Pseudo linearity of F-measure I

Objective : find a way to optimize the F-measure F_β

$$F_\beta = \frac{(1+\beta^2)(P-FN)}{(1+\beta^2)P - FN + FP} = \frac{(1+\beta^2)(P-e_1)}{(1+\beta^2)P - e_1 + e_2}.$$

Two important quantities : $e_1 = FN$ et $e_2 = FP$ linked to the empirical risk $\mathcal{R}$.

How to make the link between F_β and $\mathcal{R}$?

$\rightarrow$ **Pseudo linearity of the F-measure !**

Pseudo linearity of F-measure II

For the sake of simplicity, in the following we will denote $\mathbf{e}$ as the error profile of our vector, *i.e.*, $\mathbf{e} = (e_1, e_2) = (FN, FP)$

Thus, the F-measure can be rewritten as :

$$F_\beta = \frac{(1+\beta^2)(P - e_1)}{(1+\beta^2)P - e_1 + e_2}.$$

This formulation shows that it is enough to focus on the optimization of the error profile in order to achieve the best F-measure. We will also denote the F-measure by F rather than F_β as we do not focus on the value of the parameter β in what follows.

Pseudo linearity of F-measure III

Définition 5.1: [Pseudo Convexity rapcsk1991pseudo]

A real differentiable function f defined on an open convex set $\mathcal{C} \subset \mathbb{R}^q$ is said to be *pseudo-convex* if for every $\mathbf{e}, \mathbf{e}' \in \mathcal{C}$,

$$\langle \nabla f(\mathbf{e}), (\mathbf{e}' - \mathbf{e}) \rangle \geq 0 \implies f(\mathbf{e}') \geq f(\mathbf{e}),$$

where ∇f denotes the gradient of the function f.

The definition of pseudo-convexity is then used to define the pseudo linearity of a function.

Pseudo linearity of F-measure IV

Définition 5.2: [Pseudo Linearity]

A function f defined on an open convex $\mathcal{C}$ is said to be *pseudo-linear* if both f and $-f$ are pseudo-convex.

A first step consists in showing that the F-measure is pseudo-linear.

Proposition 5.1: Pseudo-linearity of the F-measure

The F-measure is a pseudo-linear function.

The proof is a simple computation, consider it as an exercise.

Pseudo linearity of F-measure V

The following Theorem can also characterize pseudo linear functions and provides a direct proof the pseudo linearity of the F-measure

Proposition 5.2: Pseudo-linearity

A non-constant function $F : D \to \mathbb{R}$, defined and differentiable on a open convex set $D \subset \mathbb{R}^d$ is pseudo-linear on D if and only if for all $\mathbf{e} \in D$, $(\mathbf{e}) \neq \mathbf{0}$, and $\exists \mathbf{a} : \mathbb{R} \to \mathbb{R}^{\mathbf{d}}$ and $\exists b : \mathbb{R} \to \mathbb{R}$ such that, for any t in the image of F :

$$F(\mathbf{e}) \geq t \iff \mathbf{a(t)}, \mathbf{e}\rangle + \mathbf{b(t)} \leq \mathbf{0},$$

and

$$F(\mathbf{e}) \leq t \iff \mathbf{a(t)}, \mathbf{e}\rangle + \mathbf{b(t)} \geq \mathbf{0}$$

Pseudo linearity of F-measure VI

Using this property,a result stated by [Cambini and Martein, 2009] gives a link between the F-measure and a cost-sensitive function, *i.e.*, a function which assigns weights to each class.

Proposition 5.3: [Theorem 3.3.9 from [Cambini and Martein, 2009]]

et f be a non-constant differentiable function on an open convex set $\mathcal{C} \subset \mathbb{R}^q, q > 0$. Then f is pseudo-linear on $\mathcal{C}$ if and only if the following properties hold :
(i) each of the level sets of f is the intersection of $\mathcal{C}$ with a hyper-plane ;
(ii) $\nabla f(\mathbf{e}) \neq 0$ for all $\mathbf{e} \in \mathcal{C}$.

Pseudo linearity of F-measure VII

Let us consider the set of error profiles $\{\mathbf{e} \in \mathbb{R}^2 \mid (1+\beta^2)P - e_1 + e_2 > 0\}$ (which is always the case in practice with the F-measure). Then according to the previous theorem, we rewrite (i) as follows :

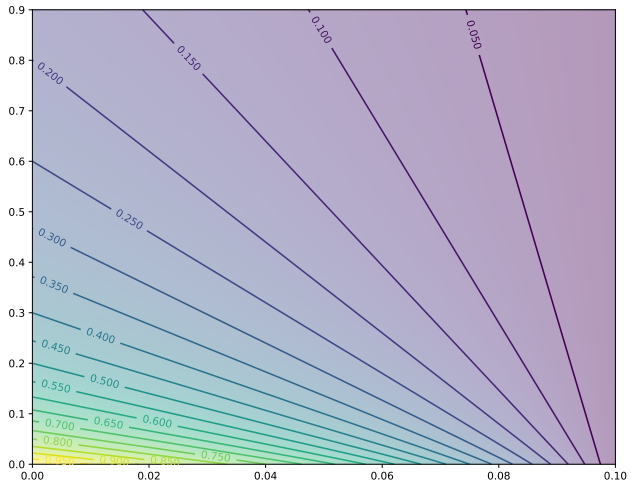
There is $\mathbf{a} : \mathbb{R} \to \mathbb{R}^{\mathbf{2}}$ and $b : \mathbb{R} \to \mathbb{R}$ such that

$$F(\mathbf{e}) = t \iff \langle \mathbf{a(t)}, \mathbf{e} \rangle + \mathbf{b(t)} = \mathbf{0},$$

which can be rewritten :

$$\langle \mathbf{a}(\mathbf{F}(\mathbf{e})), \mathbf{e} \rangle + \mathbf{b}(\mathbf{F}(\mathbf{e})) \quad = \quad 0. \tag{1}$$

Pseudo linearity of F-measure VIII



The level sets are represented in the $(e_1, e_2)-$space by the straight lines which can be seen as affine hyperplanes in a space of any dimension.

A Link with Empiricial Risk Minimization I

For the F-measure, the functions $\mathbf{a}$ and b are defined by $\mathbf{a}(\mathbf{t}) = (\mathbf{1} + \beta^{\mathbf{2}} - \mathbf{t}, \mathbf{t})$ and $b(t) = (1 + \beta^2)P(t - 1)$.

The term $\langle \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle$ can be seen as a weighted error loss function, and thus $\mathbf{a}(\mathbf{t})$ can be seen as the costs to assign to each class.

This loss function, thus its associated empirical risk will be used, essentially in the experimental part, as the quantity to minimize in order to maximize the F-measure. The link between the minimization of such a risk and the F-measure maximization is explained in the following Proposition

A Link with Empiricial Risk Minimization II

Proposition 5.4: Optimization of the F-measure

Let $F^\star = \max_{\mathbf{e}' \in \mathcal{E}(\mathcal{H})} F(e')$. We then have :

$$\mathbf{e} \in \arg\min_{\mathbf{e}' \in \mathcal{E}(\mathcal{H})} \langle \mathbf{a}(\mathbf{F}^\star), \mathbf{e}' \rangle \iff \mathbf{F}(\mathbf{e}) = \mathbf{F}^\star.$$

Proof

Let $\mathbf{e}^\star \in \arg\max_{\mathbf{e}' \in \mathcal{E}(\mathcal{H})} F(\mathbf{e}')$ and let $\mathbf{a}^\star = \mathbf{a}(\mathbf{F}(\mathbf{e}^\star)) = \mathbf{a}(\mathbf{F}^\star)$.

Using the pseudo-linearity of the F-measure, we note that the of $\mathbf{e}$ such that $\langle \mathbf{a}^\star, \mathbf{e} \rangle = \langle \mathbf{a}^\star, \mathbf{e}^\star \rangle$ corresponds to the level set

$$\{\mathbf{e} \mid F(\mathbf{e}) = F(\mathbf{e}^\star) = F^\star\}.$$

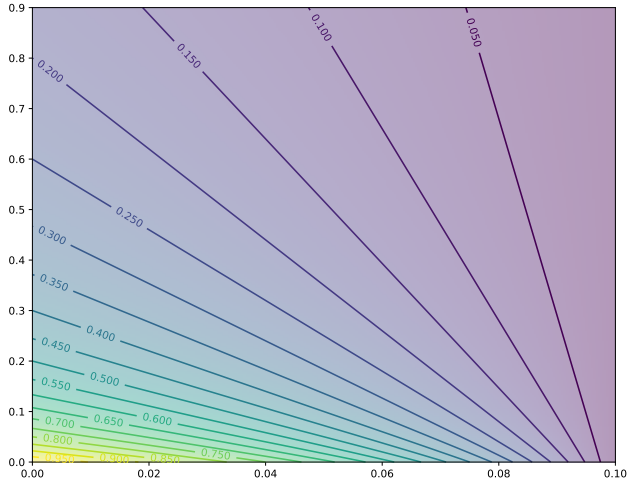
A Link with Empiricial Risk Minimization III

Thus, we only need to show that $\mathbf{e}^\star$ is a minimizer of the weighted empirical risk, *i.e.*, a minimizer of $\mathbf{e}' \mapsto \langle \mathbf{a}, \mathbf{e}' \rangle$ in $\mathcal{E}(\mathcal{H})$.
To see this, just note that the pseudo-linearity of the F-measure (or the pseudo-convexity, exercise) leads to the following implication

$$\forall \mathbf{e}' \in D,\ F(\mathbf{e}^\star) \geq F(\mathbf{e}') \implies \langle \mathbf{a}^\star, \mathbf{e}^\star \rangle \leq \mathbf{a}^\star, \mathbf{e}' \rangle.$$

This last inequality directly shows that $\mathbf{e}^\star$ is a minimizer of the empirical risk.

A Link with Empiricial Risk Minimization IV



- F_β level sets are hyperplanes in the (e_1, e_2)-space :

$$\forall t \in [0,1],\ F_\beta(\epsilon_0) = t \iff \exists\, b \text{ t.q. } \langle t), \epsilon_0 \rangle + b(t) = 0.$$

- $\mathbf{a}$: weights assigned to the errors
- $\langle \mathbf{a(t)}, \epsilon_\mathbf{0} \rangle$: weighted version of $\mathcal{R}$.

$$\rightarrow \text{Good choice of } t \iff \text{Optimizing } F_\beta.$$

Bound on the optimal F-measure I

We first present the result obtained by [Parambath et al., 2014].

Bound on the optimal F-measure II

Proposition 5.5: Bound F-measure [Parambath et al., 2014]

Let $\varepsilon_0 \geq 0$ and $\varepsilon_1 \geq 0$, and assume that there exists $\varphi > 0$ such that for all $\mathbf{e}, \mathbf{e}' \in \mathcal{E}(\mathcal{H})$ satisfying $F(\mathbf{e}') > F(\mathbf{e})$, we have :

$$F(\mathbf{e}') - F(\mathbf{e}) \leq \varphi \langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} - \mathbf{e}' \rangle.$$

Then, let us take $\mathbf{e}^\star$ a maximizer of the F-measure, and denote $\mathbf{a}^\star = \mathbf{a}(\mathbf{F}(\mathbf{e}^\star))$. Let furthermore $\mathbf{g} \in \mathbb{R}_+^d$ and $h \in \mathcal{H}$ satisfying

$$(i) \|\mathbf{g} - \mathbf{a}^\star\|_2 \leq \varepsilon_0 \quad (\mathbf{ii}) \langle \mathbf{g}, \mathbf{e} \rangle \leq \min_{\mathbf{e}' \in \mathcal{E}(\mathcal{H})} \langle \mathbf{g}, \mathbf{e}' \rangle + \varepsilon_1.$$

We have : $F(\mathbf{e}^\star) \leq F(\mathbf{e}) + \varphi(2\varepsilon_0 M + \varepsilon_1), \quad M = \max_{\mathbf{e}' \in \mathcal{E}(\mathcal{H})} \|\mathbf{e}'\|_2.$

Bound on the optimal F-measure III

We are not going to show directly this result since in cannot be used like this, but are going to use another formulation of the later which is more suitable for interpretation.

Bound on the optimal F-measure IV

Proposition 5.6: An improved bound

Let $t, t' \in [0,1]$ and $\varepsilon_1 \geq 0$. Suppose that there is $\Phi > 0$ such that for all $\mathbf{e}, \mathbf{e}' \in \mathcal{E}(\mathcal{H})$ satisfying $F(\mathbf{e}') > F(\mathbf{e})$, we have :

$$F(\mathbf{e}') - F(\mathbf{e}) \geq \Phi \langle \mathbf{a}(\mathbf{t}'), \mathbf{e} - \mathbf{e}' \rangle. \tag{2}$$

Furthermore, suppose that we have the two following conditions

$$(i) \quad \|\mathbf{a}(\mathbf{t}) - \tilde{(}\mathbf{t}')\|_2 \leq 2\left|\mathbf{t} - \mathbf{t}'\right|, \qquad (\mathbf{ii}) \ \langle \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle \leq \min_{\mathbf{e}'' \in \mathcal{E}(\mathcal{H})} \langle \mathbf{a}(\mathbf{t}), \mathbf{e}'' \rangle + \varepsilon_1.$$

Let us also set $M = \max_{\mathbf{e}'' \in \mathcal{E}(\mathcal{H})} \|\mathbf{e}''\|_2$, then we have :

$$F(\mathbf{e}') \leq F(\mathbf{e}) + \Phi\varepsilon_1 + 4M\Phi\left|t' - t\right|.$$

Bound on the optimal F-measure V

According to the authors, the point (i) is a consequence of $\mathbf{a}$ of being Lipschitz continuous with Lipschtiz constant equal to 2. The point (ii) is just the expression of the sub-optimality of the learned classifier.

Bound on the optimal F-measure VI

Proof

For all $\mathbf{e}, \tilde{\mathbf{e}} \in \mathcal{E}(\mathcal{H})$ and $t, t' \in [0,1]$, we have :

$$\begin{aligned}\langle \mathbf{a(t)}, \tilde{\mathbf{e}} \rangle &= \underbrace{\langle \mathbf{a(t)} - \mathbf{a(t')}, \tilde{\mathbf{e}} \rangle} + \langle \mathbf{a(t')}, \tilde{\mathbf{e}} \rangle, \\ &\leq \langle \mathbf{a(t')}, \tilde{\mathbf{e}} \rangle + \mathbf{2M}\left|\mathbf{t'} - \mathbf{t}\right|,\end{aligned}$$

where we have successively applied the Cauchy-Schwarz inequality and (i). Then :

$$\min_{\mathbf{e''} \in \mathcal{E}(\mathcal{H})} \langle \mathbf{a(t)}, \mathbf{e''} \rangle \leq \min_{\mathbf{e''} \in \mathcal{E}(\mathcal{H})} \langle \mathbf{a(t')}, \mathbf{e''} \rangle + \mathbf{2M}\left|\mathbf{t'} - \mathbf{t}\right| = \langle \mathbf{a(t')}, \mathbf{e'} \rangle + \mathbf{2M}\left|\mathbf{t'} - \mathbf{t}\right| \tag{3}$$

where $\mathbf{e'}$ denotes the error profile learned by the optimal classifier trained with the cost function $\mathbf{a(t')}$ and is such that $F(\mathbf{e'}) > F(\mathbf{e})$.

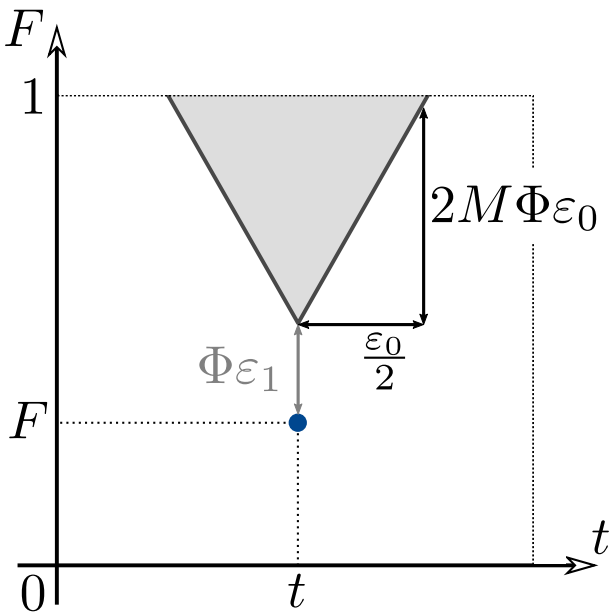
Bound on the optimal F-measure VII

Then, writing $\langle \mathbf{a}(\mathbf{t}'), \mathbf{e} \rangle = \langle \mathbf{a}(\mathbf{t}') - \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle + \langle \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle$ and applying the Cauchy-Schwarz inequality, we have :

$$
\begin{aligned}
\langle \mathbf{a}(\mathbf{t}'), \mathbf{e} \rangle &\leq \underbrace{\langle \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle} + \mathbf{2M}\left|\mathbf{t}' - \mathbf{t}\right|, \\
&\qquad \downarrow \text{ using the fact that the learned classifier is sub-optimal} \\
&\leq \min_{\mathbf{e}'' \in \mathcal{E}(\mathcal{H})} \langle \mathbf{a}(\mathbf{t}), \mathbf{e}'' \rangle + \varepsilon_{\mathbf{1}} + 2M\left|t' - t\right|, \\
&\leq \langle \mathbf{a}(\mathbf{t}'), \mathbf{e}' \rangle + \varepsilon_{\mathbf{1}} + \mathbf{4M}\left|\mathbf{t}' - \mathbf{t}\right|,
\end{aligned}
$$

where the second inequality comes from (ii) and the last inequality comes from Equation (3). By plugging this last inequality in Inequality (2), we get the result.
Furthermore, the existence of the constant Φ will be shown later.

Bound on the optimal F-measure VIII



Interpretation existing bound

of [Parambath et al., 2014]

$$F(\epsilon_0') \leq F(\epsilon_0) + \Phi \cdot (2\varepsilon_0 M + \varepsilon_1)$$

$$F(\epsilon_0') \leq F(\epsilon_0) + \Phi\varepsilon_1 + 4M\Phi|t' - t|.$$

où

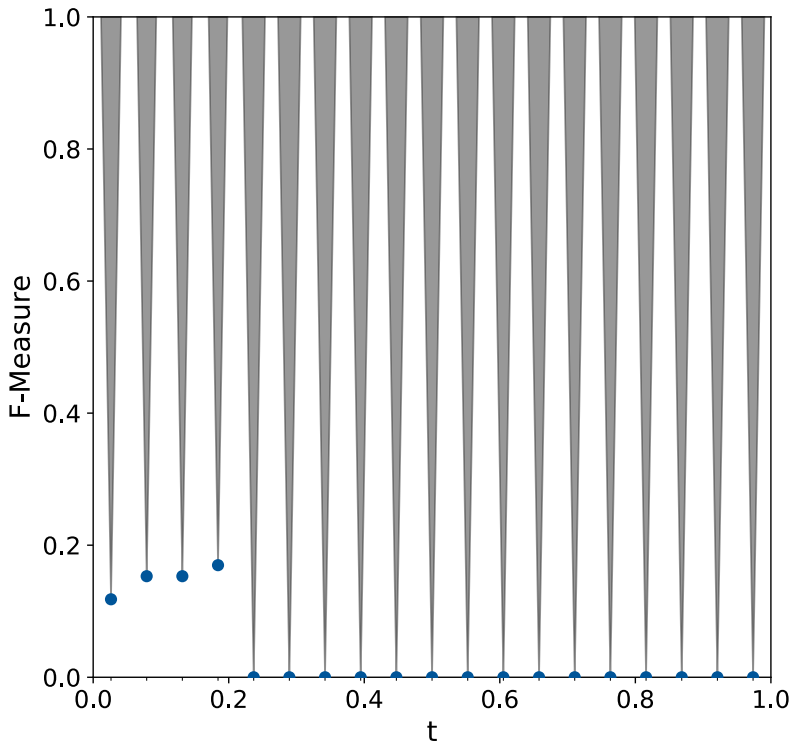
- ε_0 : distance to optimal weights
- $M = \max_{\epsilon_0''} \|\epsilon_0''\|_2$
- $\Phi = (\beta^2 P)^{-1}$ independent from ϵ_0

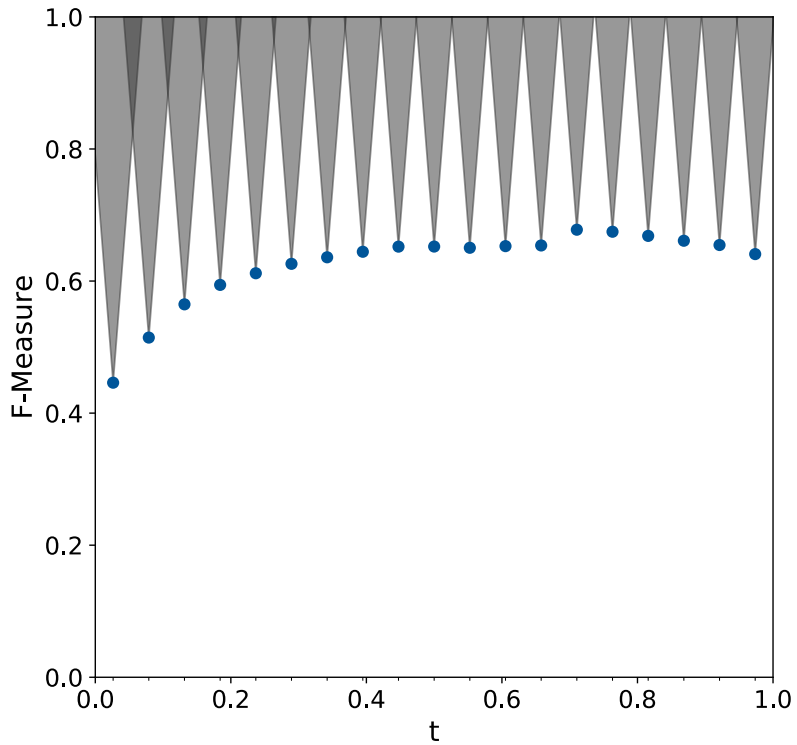
Bound on the optimal F-measure IX

In practice, since the function $\mathbf{a} : [\mathbf{0}, \mathbf{1}] \to \mathbb{R}^{\mathbf{d}}$, which assigns costs to probabilities of error, is Lipschitz-continuous with constant 2, it is sufficient to discretize the interval $[0, 1]$ to have a set of evenly spaced values $\{t_1, ..., t_C\}$, *i.e.*, $t_{j+1} - t_j = \varepsilon_0/2$ to obtain an ε_0-cover of the possible costs.

Using the approximate guarantee of their result, learning a cost-sensitive classifier for each costs and selecting the one with optimal F-measure a posteriori is sufficient to obtain a $M(2\varepsilon_0 + \varepsilon_1)$-optimal solution.

Bound on the optimal F-measure X





Bound on the optimal F-measure XI

This algorithm present one advantage, it gives the possibility to run several model in parallel and then choose the appropriate one, *i.e.* the one for which the F-measure is maximized.

However, it requires to lunch several model in order to achieve a sufficiently precise model. We can then wonder if it is possible to achieve similar results but with a limited budget, in other words, by learning a limited number of models.

This is what we are going to explore now, using a similar idea as the one developed by the authors.

CONE : an Algorithm for F-measure Optimization I

We aim to study the importance of the function $\mathbf{a}$ and of the parameter t to characterize the difference of F-measures between any two error profiles.

Step 1 : impact of a change in the error profile

We first derive the relation between the difference in F-measures and the difference in error profiles.
Let $\mathbf{e}$ and $\mathbf{e}'$ be any two error profiles and $F(\mathbf{e})$ and $F(\mathbf{e}')$ the corresponding F-measures.
From the pseudo-linearity property (Equation (1)), we have :

$$0 = \langle \mathbf{a}(\mathbf{F}(\mathbf{e})), \mathbf{e} \rangle + \mathbf{b}(\mathbf{F}(\mathbf{e})), \quad (4)$$

$$0 = \langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e}' \rangle + \mathbf{b}(\mathbf{F}(\mathbf{e}')). \quad (5)$$

CONE : an Algorithm for F-measure Optimization II

We now develop $\langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} - \mathbf{e}' \rangle$ and make the difference in F-measures appears in its expression.

$$\begin{aligned}
\langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} - \mathbf{e}' \rangle = & \quad \langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} \rangle + \mathbf{b}(\mathbf{F}(\mathbf{e}')), \\
& \downarrow \text{ using Equation (4)} \\
= & \quad \underbrace{\langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} \rangle - \langle \mathbf{a}(\mathbf{F}(\mathbf{e})), \mathbf{e} \rangle}_{} \underbrace{- \mathbf{b}(\mathbf{F}(\mathbf{e})) + \mathbf{b}(\mathbf{F}(\mathbf{e}'))}_{}, \\
& \downarrow \text{ using the linearity of the inner product and } b \\
= & \quad \underbrace{\langle \mathbf{a}(\mathbf{F}(\mathbf{e}')) - \mathbf{a}(\mathbf{F}(\mathbf{e})), \mathbf{e} \rangle}_{} + P(1+\beta^2)(F(\mathbf{e}') - F(\mathbf{e})) \\
& \downarrow \text{ using the definition of } \mathbf{a} \\
= & \quad (\mathbf{e}_2 - \mathbf{e}_1)(F(\mathbf{e}') - F(\mathbf{e})) + P(1+\beta^2)(F(\mathbf{e}') - F(\mathbf{e}) \\
& \downarrow \text{ factorizing} \\
\langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} - \mathbf{e}' \rangle = & \quad \big(F(\mathbf{e}') - F(\mathbf{e})\big) \cdot \big((1+\beta^2)P - e_1 + e_2\big) \,,
\end{aligned}$$

CONE : an Algorithm for F-measure Optimization III

where the second line uses Equation (4). The third one uses the linearity of the inner product and the definition of b. The fourth one uses Equation (4) and the last line uses the definition of $\mathbf{a}$ and b defined previously.

Now we can rewrite the difference in F-measures as :

$$F(\mathbf{e}') - F(\mathbf{e}) \quad = \quad \Phi_{\mathbf{e}} \cdot \langle \mathbf{a}(\mathbf{F}(\mathbf{e}')), \mathbf{e} - \mathbf{e}' \rangle, \tag{6}$$

where $\Phi_{\mathbf{e}} = \dfrac{1}{(1+\beta^2)P - e_1 + e_2}$.

CONE : an Algorithm for F-measure Optimization IV

Step 2 : a first bound on the F-measure $F(\mathbf{e}')$

We suppose that we have a value of t for which a weighted-classifier with weights $\mathbf{a(t)}$ has been learned. This classifier has an error profile $\mathbf{e}$ and a F-measure $F(\mathbf{e})$.

We now imagine a hypothetical classifier which leads to an error profile $\mathbf{e}^\star$ the optimal error profile, i.e. the one that maximized the F-measure and for which we have $F(\mathbf{e}^\star) = t^\star$.

Starting from the result obtained in Equation (6), we have :

CONE : an Algorithm for F-measure Optimization V

$$
\begin{aligned}
F(\mathbf{e}^\star) - F(\mathbf{e}) = \quad & \Phi_{\mathbf{e}} \left(\underbrace{\langle \mathbf{a}(\mathbf{t}^\star), \mathbf{e} \rangle} - \langle \mathbf{a}(\mathbf{t}^\star), \mathbf{e}^\star \rangle \right), \\
& \downarrow \text{ using } \mathbf{a}' = \mathbf{a} + \mathbf{a}^\star - \mathbf{a} \\
= \quad & \Phi_{\mathbf{e}} \left(\langle \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle + \underbrace{\langle \mathbf{a}(\mathbf{t}^\star) - \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle} - \langle \mathbf{a}(\mathbf{t}^\star), \mathbf{e}^\star \rangle \right), \\
& \downarrow \text{ using the defintion of } \mathbf{a} \\
= \quad & \Phi_{\mathbf{e}} \left(\underbrace{\langle \mathbf{a}(\mathbf{t}), \mathbf{e} \rangle} + (\mathbf{t}^\star - \mathbf{t})(\mathbf{e_2} - \mathbf{e_1}) - \langle \mathbf{a}(\mathbf{t}^\star), \mathbf{e}^\star \rangle \right), \\
& \downarrow \text{ introducing the sub-optimality of the learned classifier} \\
\leq \quad & \Phi_{\mathbf{e}} \left(\underbrace{- \langle \mathbf{a}(\mathbf{t}^\star), \mathbf{e}^\star \rangle + \langle \mathbf{a}(\mathbf{t}), \mathbf{e}^\star \rangle} + \varepsilon_1 + (\mathbf{t}^\star - \mathbf{t})(\mathbf{e_2} - \mathbf{e_1}) \right) \\
& \downarrow \text{ using the definition of } \mathbf{a}
\end{aligned}
$$

CONE : an Algorithm for F-measure Optimization VI

$$
\begin{aligned}
& \leq \quad \Phi_{\mathbf{e}} \left((t - t^\star)(e_2^\star - e_1^\star) + \varepsilon_1 + (t^\star - t)(e_2 - e_1) \right), \\
F(\mathbf{e}^\star) - F(\mathbf{e}) \leq & \quad \Phi_{\mathbf{e}} \varepsilon_1 + \Phi_{\mathbf{e}} \cdot (e_2 - e_1 - (e_2^\star - e_1^\star))(t^\star - t). \qquad (7)
\end{aligned}
$$

We have successively used the linearity of the inner product, introduced $\mathbf{a(t)}$ and its definition in the first three equalities.
The first inequality uses $\langle \mathbf{a(t)}, \mathbf{e} \rangle \leq \langle \mathbf{a(t)}, \mathbf{e_{best}} \rangle + \varepsilon_\mathbf{1}$, the sub-optimality of the $\mathbf{a(t)}$-weighted-error classifier.

CONE : an Algorithm for F-measure Optimization VII

The value of ε_1 represents the excess of risk of the classifier which aims to minimize the $\mathbf{a}(\mathbf{t})$-weighted-error.

More precisely, it represents the difference of risks between our classifier and the best classifier h_{best} (in terms of $\mathbf{a}(\mathbf{t})$-weight-error) in our set of hypotheses $\mathcal{H}$.

CONE : an Algorithm for F-measure Optimization VIII

We are interested in upper bounding the optimal value of the F-measure using equation (7).

For this purpose, we consider any possible value of t', for which we learn a hypothetical classifier with weights $\mathbf{a}(\mathbf{t}')$, that gives us an error profile $\mathbf{e}'$ and a F-measure $F(\mathbf{e}')$. If t' happens to be the optimal value which, by weighting the errors with $\mathbf{a}(\mathbf{t}')$, maximizes the F-measure then $F(\mathbf{e}') = t'$ for $\mathbf{e}' = \underset{\hat{\mathbf{e}} \in \mathcal{E}(\mathcal{H})}{\text{argmin}} \langle \mathbf{a}(\mathbf{t}'), \hat{\mathbf{e}} \rangle$.
Equation (7) can then be applied and gives :

$$F(\mathbf{e}') - F(\mathbf{e}) \leq \quad \Phi_{\mathbf{e}} \varepsilon_1 + \Phi_{\mathbf{e}} \cdot (e_2 - e_1 - (e'_2 - e'_1))(t' - t)$$

CONE : an Algorithm for F-measure Optimization IX

Step 3 : a tight bound on $\mathbf{e}'_2 - \mathbf{e}'_1$

To complete this bound we need to study the difference $e'_2 - e'_1$. We aim to bound this difference according to the sign of $t - t'$ and under the constraint that $\mathbf{e}'$ is such that $F(\mathbf{e}') > F(\mathbf{e})$ in order to have a tighter bound.

We first need a preliminary result to explain if we have to give an upper bound or a lower bound on $e'_2 - e'_1$.

Lemme 5.1:

The difference $(e_1 - e_2)(t)$ is increasing when $\mathbf{e}(t)$ is obtained from an optimal weighted-classifier trained with costs $\mathbf{a}(\mathbf{t})$.

CONE : an Algorithm for F-measure Optimization X

Proof

Let $t > t'$, $\mathbf{e}(t)$ and $\mathbf{e}(t')$ the vector of missclassified examples obtained with an optimal classifier (in the Bayes sense) trained with costs $\mathbf{a}(\mathbf{t})$ and $\mathbf{a}(\mathbf{t}')$ respectively.

We thus have the following inequalities :

$$t \cdot e_2(t) + (1 + \beta^2 - t)\, e_1(t) \leq t \cdot e_2(t') + (1 + \beta^2 - t)\, e_1(t'),$$

and

$$t' \cdot e_2(t') + (1 + \beta^2 - t')\, e_1(t') \leq t' \cdot e_2(t) + (1 + \beta^2 - t')\, e_1(t).$$

CONE : an Algorithm for F-measure Optimization XI

By multiplying the second equation by -1 and summing the two equations, we get :

$$(t - t')(e_1(t) - e_2(t) \geq (t - t')(e_1(t') - e_2(t')).$$

Thus :

$$e_1(t') - e_2(t') \leq e_1(t) - e_2(t).$$

CONE : an Algorithm for F-measure Optimization XII

According to this Lemma the difference $(e_1 - e_2)(t)$ is an increasing function of t, thus, when $t' < t$, $e_2 - e_1 - (e'_2 - e'_1) < 0$ and $e'_2 - e'_1$ shall be maximized to have a tight bound that preserves this inequality.

Similarly, when $t' > t$, we have to minimize the difference $e'_2 - e'_1$.

Maximizing (resp. minimizing) this difference consists of a constraint optimization problem.

The following Lemma gives the analytical solution of the two optimization problems.

CONE : an Algorithm for F-measure Optimization XIII

Lemme 5.2: An optimal bound on the error profile

Let $\mathbf{e}$ be the error profile obtained with an hypothesis h learned with the cost function $\mathbf{a}(\mathbf{t})$. Let $\mathbf{e}'$ be an error profile such that $F(\mathbf{e}') > F(\mathbf{e})$. Then the difference $e'_2 - e'_1$ is equal to :

$$M_{\max} = \max_{\substack{\mathbf{e}' \in \mathcal{E}(\mathcal{H}) \\ s.t.\ F(\mathbf{e}') > F(\mathbf{e})}} (e'_2 - e'_1) = e_2 + \min\left(N - e_2, \frac{e_1(\beta^2 P + e_2)}{P - e_1}\right),\ t' < t,$$

and

$$M_{\min} = \min_{\substack{\mathbf{e}' \in \mathcal{E}(\mathcal{H}) \\ s.t.\ F(\mathbf{e}') > F(\mathbf{e})}} (e'_2 - e'_1) = -e_1 - \frac{e_2(P - e_1)}{\beta^2 P + e_2},\ t' > t.$$

CONE : an Algorithm for F-measure Optimization XIV

Step 4 : bounds on the F-measure $F(\mathrm{e}')$

It remains to combine the different results in order to reach the following theorem.

CONE : an Algorithm for F-measure Optimization XV

Proposition 5.7:

Let $\mathbf{e}$ be the error profile obtained with a classifier trained with the parameter t and $F(\mathbf{e})$ its associated F-measure value. Let us also consider $\Phi_{\mathbf{e}}$ as defined in Equation (6) and $\varepsilon_1 > 0$ the sub-optimality of our linear classifier. Then for all $t' < t$:

$$F(\mathbf{e}') \leq F(\mathbf{e}) + \Phi_{\mathbf{e}}\varepsilon_1 + \Phi_{\mathbf{e}} \cdot (e_2 - e_1 - M_{max})(t' - t),$$

where $M_{\text{max}} = \max_{\substack{\mathbf{e}'' \in \mathcal{E}(\mathcal{H}) \\ s.t.\ F(\mathbf{e}'') > F(\mathbf{e})}} (e_2'' - e_1'')$

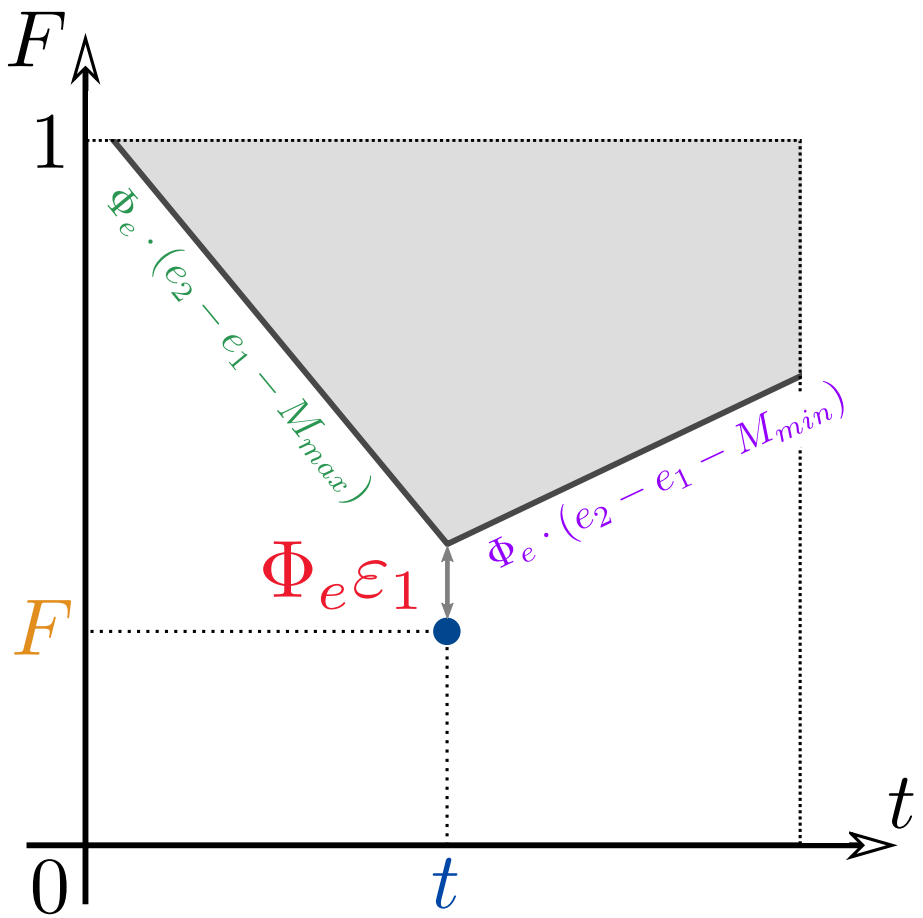
and, for all $t' > t$:

$$F(\mathbf{e}') \leq F(\mathbf{e}) + \Phi_{\mathbf{e}}\varepsilon_1 + \Phi_{\mathbf{e}} \cdot (e_2 - e_1 - M_{min})(t' - t),$$

where $M_{\text{min}} = \min_{\substack{\mathbf{e}'' \in \mathcal{E}(\mathcal{H}) \\ s.t.\ F(\mathbf{e}'') > F(\mathbf{e})}} (e_2'' - e_1'')$.

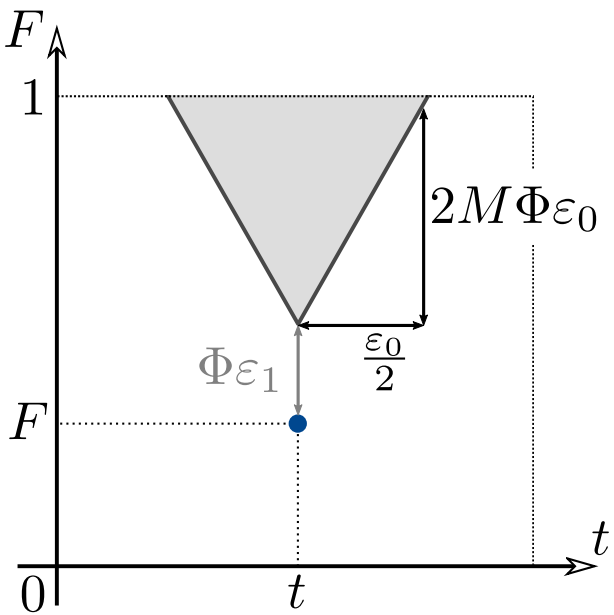
CONE : an Algorithm for F-measure Optimization

An asymmetric cone



CONE : an Algorithm for F-measure Optimization

Existing results



Interpretation existing bound

of [Parambath et al., 2014]

$$F(\epsilon_0') \leq F(\epsilon_0) + \Phi \cdot (2\varepsilon_0 M + \varepsilon_1)$$

$$F(\epsilon_0') \leq F(\epsilon_0) + \Phi\varepsilon_1 + 4M\Phi|t' - t|.$$

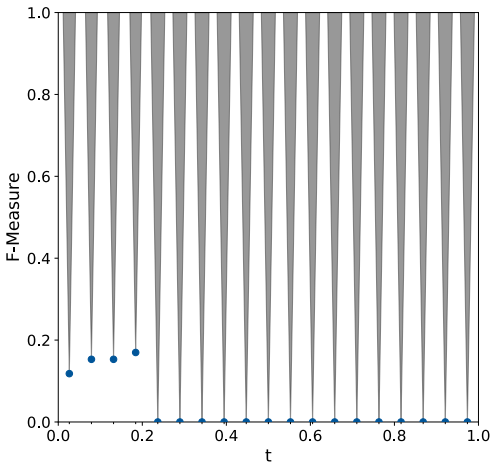
où

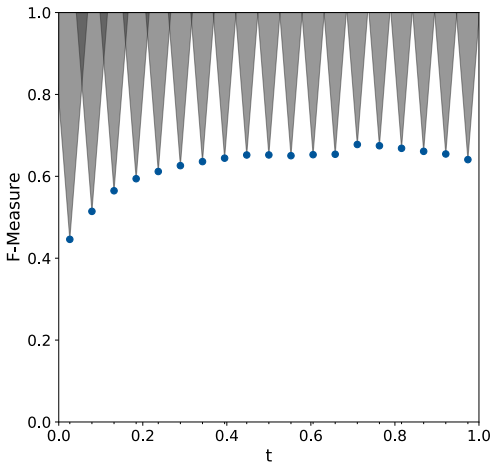
- ε_0 : distance to optimal weights
- $M = \max_{\epsilon_0''} \|\epsilon_0''\|_2$
- $\Phi = (\beta^2 P)^{-1}$ independent from ϵ_0

CONE : an Algorithm for F-measure Optimization

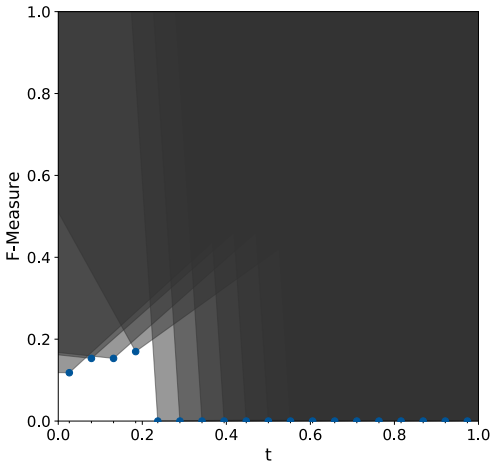
Bounds comparison : [Parambath et al., 2014] vs Improved

Similar bounds ...

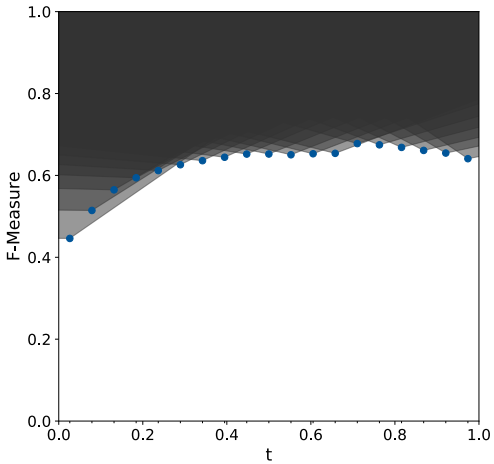




Abalone 12



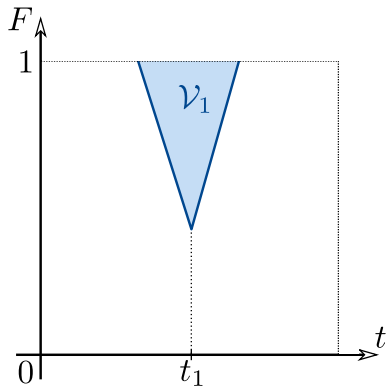
Adult



... with highly different slopes.

CONE : an Algorithm for F-measure Optimization

An iterative algorithm



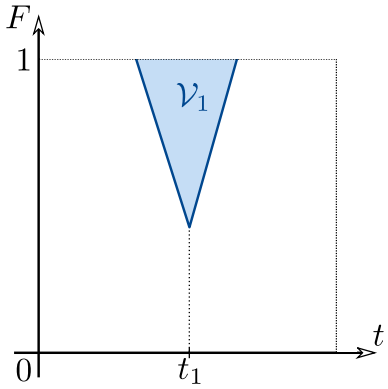
Step 1

Step 1 : Take the middle of the t-space of reasearch : $t_1 = 0.5$

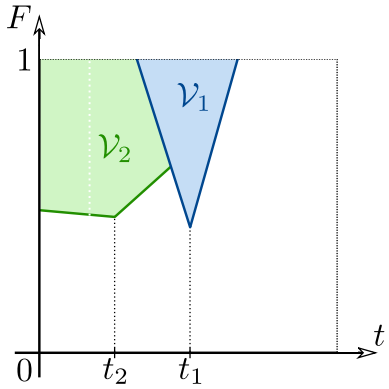
$\rightarrow$ Highest values of F in $[0, t_1]$

CONE : an Algorithm for F-measure Optimization

An iterative algorithm



Step 1



Step 2

Step 1 : Take the middle of the t-space of reasearch : $t_1 = 0.5$

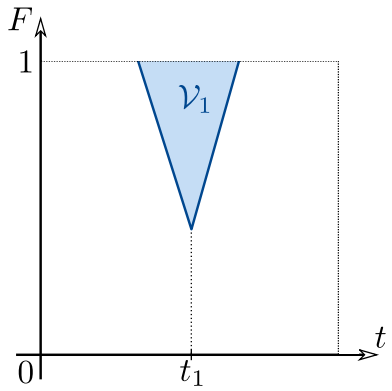
$\rightarrow$ Highest values of F in $[0, t_1]$

Step 2 : Choose t_2 in the middle of $[0, t_1]$

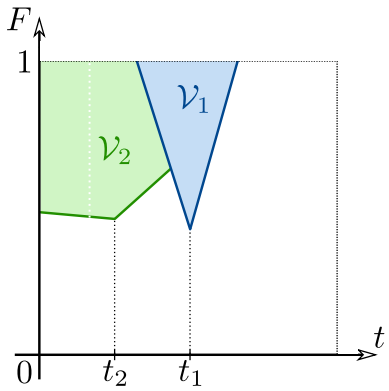
$\rightarrow$ Highest values of F in $[t_1, 1]$

CONE : an Algorithm for F-measure Optimization

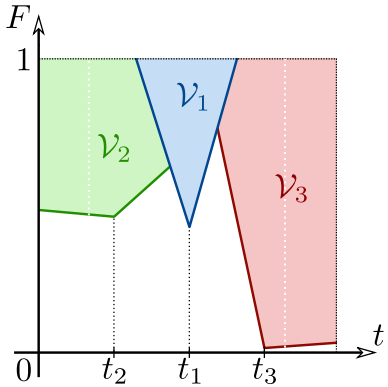
An iterative algorithm



Step 1



Step 2



Step 3

Step 1 : Take the middle of the t-space of reasearch : $t_1 = 0.5$

$\rightarrow$ Highest values of F in $[0, t_1]$

Step 2 : Choose t_2 in the middle of $[0, t_1]$

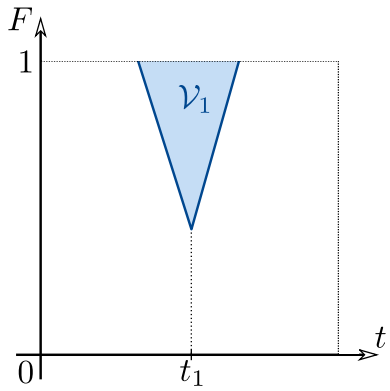
$\rightarrow$ Highest values of F in $[t_1, 1]$

Step 3 : Choose t_3 in the middle of $[t_1, 1]$

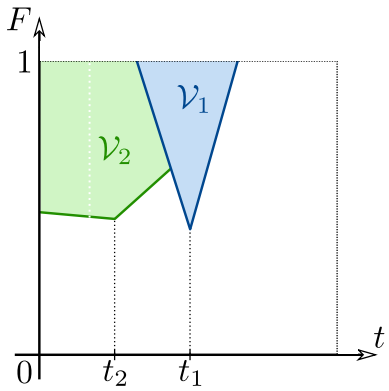
$\rightarrow$ Highest values of F in $[t_1, t_3]$

CONE : an Algorithm for F-measure Optimization

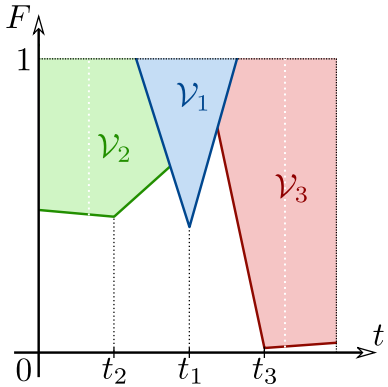
An iterative algorithm



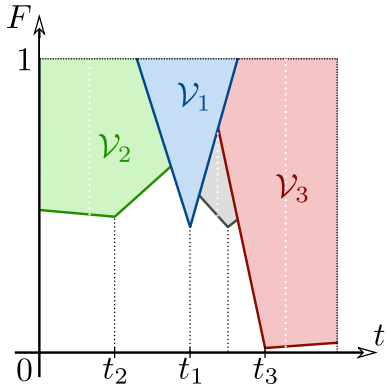
Step 1



Step 2



Step 3



Step 4

Step 1 : Take the middle of the t-space of reasearch : $t_1 = 0.5$

$\rightarrow$ Highest values of F in $[0, t_1]$

Step 2 : Choose t_2 in the middle of $[0, t_1]$

$\rightarrow$ Highest values of F in $[t_1, 1]$

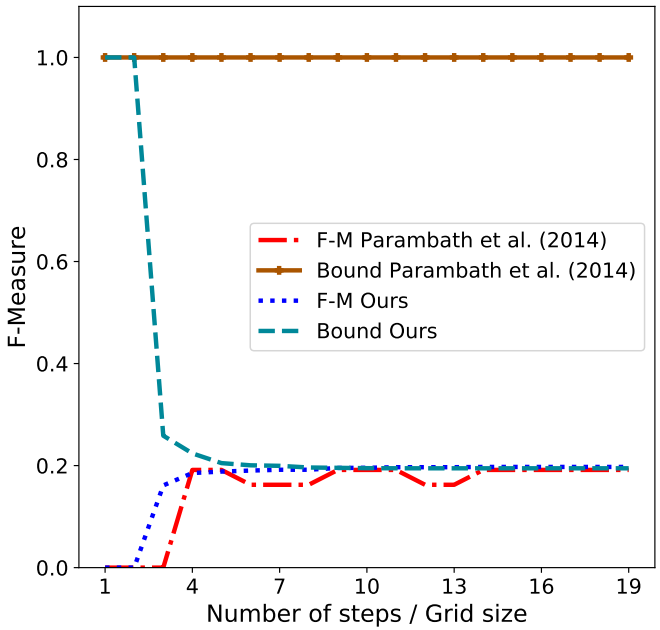
Step 3 : Choose t_3 in the middle of $[t_1, 1]$

$\rightarrow$ Highest values of F in $[t_1, t_3]$

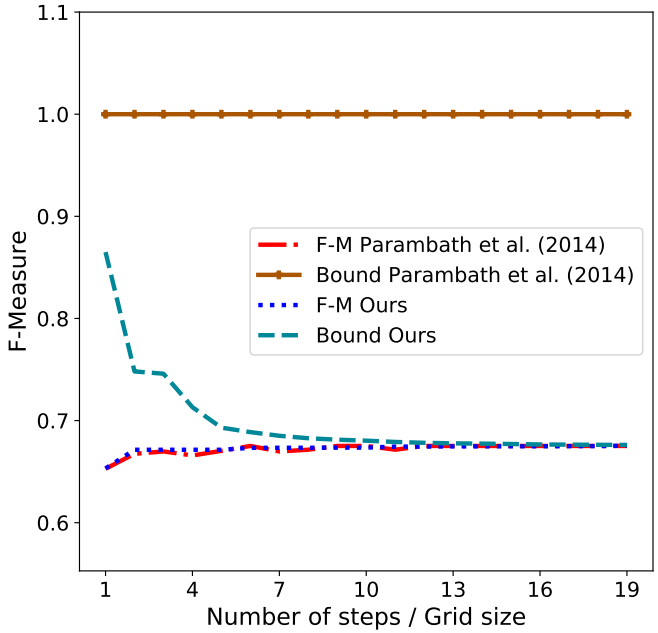
Step 4 : Choose t_4 in the middle of $[t_1, t_3]$

CONE : an Algorithm for F-measure Optimization

Comparison in terms of convergence



Abalone 12



Adult

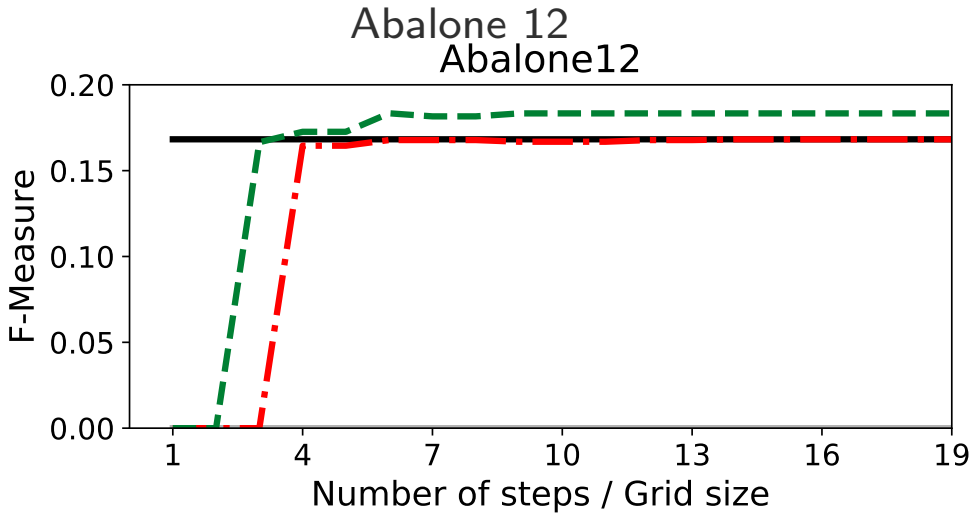
$\longrightarrow$ A more informative bound

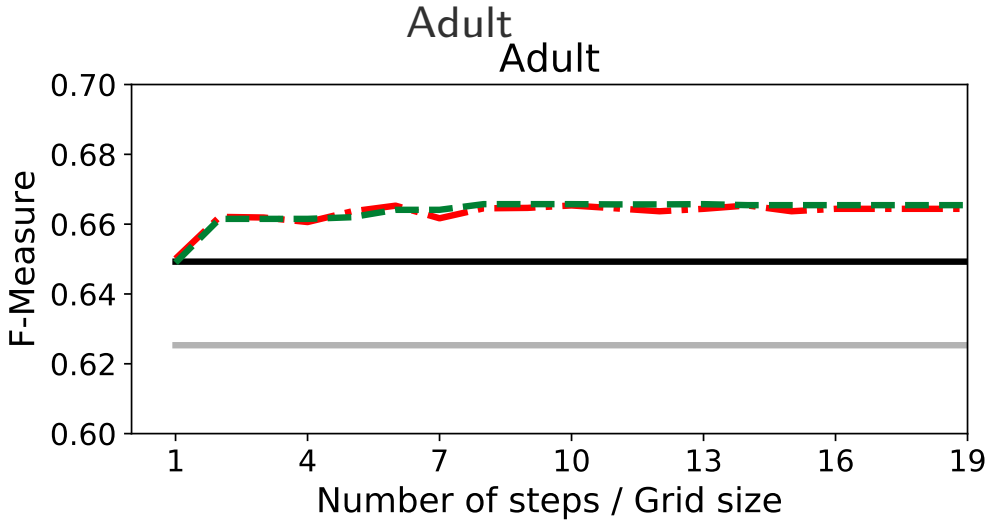
$\longrightarrow$ A faster convergence

$\longrightarrow$ Improves the performances

CONE : an Algorithm for F-measure Optimization

Comparison of performances





SVM : a linear SVM

SVM$_{I.R}$: a linear SVM with weighted errors

SVM$_G$: grid approach parambath2014optimizing

SVM$_C$: improved approach

References I


Cambini, A. and Martein, L. (2009).
Generalized Convexity and Optimization : Theory and Applications.
Lecture Notes in Economics and Mathematical Systems 616.
Springer-Verlag Berlin Heidelberg, 1 edition.


Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002).
Smote : Synthetic minority over-sampling technique.
Journal of Artificial Intelligence Research, 16(1) :321–357.


Fernández, A., Garcia, S., Herrera, F., and Chawla, N. V. (2018).
Smote for learning from imbalanced data : Progress and challenges, marking the 15-year anniversary.
Journal of Artificial Intelligence Research, 61 :863–905.

References II


Hart, P. (1968).
The condensed nearest neighbor rule.
IEEE Transactions on Information Theory, 14(3) :515–516.


Parambath, S. P., Usunier, N., and Grandvalet, Y. (2014).
Optimizing f-measures by cost-sensitive classification.
In *Advances in Neural Information Processing Systems (NIPS-14)*, pages 2123–2131.


Tomek, I. (1976).
Two modifications of cnn.
IEEE Trans. Systems, Man and Cybernetics, 6 :769–772.

References III


Wilson, D. L. (1972).
Asymptotic properties of nearest neighbor rules using edited data.
IEEE Transactions on Systems, Man, and Cybernetics,
SMC-2(3) :408–421.