Distortions and Learning

Richard Nock

Centre d'**E**tudes et de **R**echerches en **E**conomie, **G**estion, **M**athématiques et **I**nformatique **A**ppliquée

Université des Antilles-Guyane
Richard.Nock@martinique.univ-ag.fr

May 2007

Introduction

Talk aims at showing **some ties** that (Computational) Geometr*ies* share with machine learning:

1. some are obvious (3% of the talk),
2. others are related to some of the most fundamental questions of machine learning:

   What is learning ? How can an Algorithm "learn" ? (97%)

Geometries generally non Euclidean, basically non Riemannian.

Slides available at
`http://www.univ-ag.fr/~rnock/Slides/Geotopal07/`

Outline

1. (What is) *Supervised Learning* ?
2. Learning Strategies
3. **Boosting** algorithms
4. Distortions for Learning: Bregman Divergences
5. Axiomatization (of Supervised Learning)
6. Minimization Algorithms
7. Related Questions and Perspectives

Supervised Learning

1. What is Supervised learning ?
2. Key components, quantities ?
3. Formal models of Supervised learning ?

Supervised Learning > Example

Tic-tac-toe endgame configurations, x vs o, x has played first.

X		O
X	X	X
O		O

wins for x

X		X
O	O	O
	X	

,

X	O	X
X	O	O
O	X	X

do not win for x

| Input | Output |

Problem

Suppose you do not know the game.
(How) can you infer an ***accurate function*** *Input* $\longrightarrow$ *Output ?*

Example: $(\boldsymbol{x}, y)$

Ground information: each endgame configuration.

X		O
X	X	X
O		O

wins for X

is represented by: $\underbrace{(\underbrace{\texttt{XBOXXXOBO}}_{\text{observation}},\ \underbrace{+1}_{\text{class}})}_{\text{example}}$

Remarks:

1. Label = synonym of class ($+1$ = wins for X; -1 = does not win for X)
2. We assume two classes wlog

Domain: $\mathcal{X}$

Let $\mathcal{X}$ denote the set of all board configurations (over $n = 9$ description variables). Some are endgames, some are not. Some are even not admissible for tic-tac-toe (XXXXXXXXX).

Distributions: D

Example are drawn i.i.d., from some fixed and unknown distribution D over $\mathcal{X} \times \{-1, +1\}$. Let weight vector $\boldsymbol{w}_1 = \hat{D}$ over $\mathcal{S}$.

Learning Sample: $\mathcal{S}$

Suppose someone takes m endgame configurations (among the 958 distinct possible, according to D) and labels them. Let

$$\mathcal{S} \stackrel{\text{def}}{=} \{(\boldsymbol{x}_i, y_i^*) : \boldsymbol{x}_i \in \mathcal{X}, y_i^* \in \{-1, +1\}\}_{i=1}^m$$

be this set ($+1$ = wins for X, -1 = does not win for X).

Classifier: H

Any function $H : \mathcal{X} \rightarrow \mathbb{S} \subseteq \mathbb{R}$. For example $\mathbb{S} = \{-1, +1\}, \mathbb{R}, \ldots$.

$$\boldsymbol{x} \in \mathcal{X} \stackrel{\text{def}}{=} \left(\begin{array}{|c|c|c|} \hline v_1 & v_2 & v_3 \\ \hline v_4 & v_5 & v_6 \\ \hline v_7 & v_8 & v_9 \\ \hline \end{array} \right)$$

Example 1 Rule $H_1(\boldsymbol{x}) \stackrel{\text{def}}{=}$ **If** $(v_1 = \mathbf{X}) \wedge (v_5 = \mathbf{X}) \wedge (v_9 = \mathbf{X})$ **Then** $+1$ **Else** -1

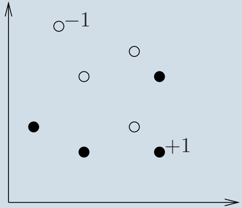
Example 2 Rule $H_2(\boldsymbol{x}) \stackrel{\text{def}}{=}$ **If** $(v_2 = \mathbf{X}) \wedge (v_5 = \mathbf{O}) \wedge (v_9 = \mathbf{X})$ **Then** -1 **Else** $+1$

Example 3 Function $H_3(\boldsymbol{x}) \stackrel{\text{def}}{=} H_1(\boldsymbol{x})H_2(\boldsymbol{x})$

Example 4 Function $H_4(\boldsymbol{x}) \stackrel{\text{def}}{=} sign(3H_1(\boldsymbol{x}) - 2H_2(\boldsymbol{x}))$
($sign(z) = +1$ iff $z \geq 0$, and -1 otherwise)

$\mathcal{H}$ = set of classifiers sharing the same model.

Geometric domain

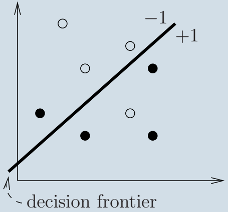
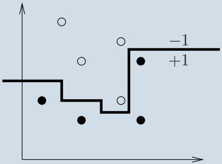
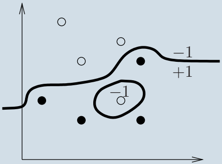


$\mathcal{X} = \mathbb{R}^2$

$|\mathcal{S}| = 8$

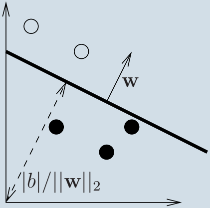
4 positive, 4 negative

Which classifier ?

		
decision frontier:		
a line	piecewise linear	curved, regions not connex

Which **set of classifiers** $\mathcal{H}$ is the best ? most popular ?

Classifier: $(\mathcal{H} =)LS$



Linear Separators (linear models)

$$H(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \in \mathbb{R}$$

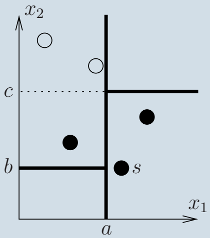
Decision frontier: $\{\boldsymbol{x} : H(\boldsymbol{x}) = 0\}$, an hyperplane

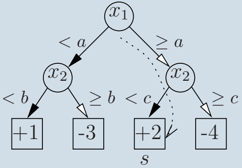
Needs $\mathcal{X} \subseteq \mathbb{R}^n$, but not a constraint: find **feature vector $\boldsymbol{f}$** s.t.

$$f_i \quad : \mathcal{X} \rightarrow \mathbb{R} \quad ,$$

and build $H(\boldsymbol{x}) = \langle \alpha, \boldsymbol{f}(\boldsymbol{x}) \rangle + \beta$.

Classifier: *DT*





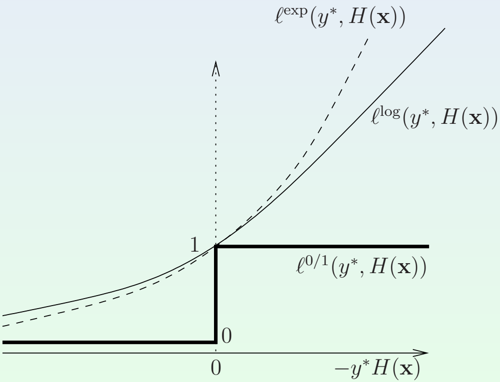
Decision Trees

Decision frontier: (portions of) axis-$\perp$ hyperplanes.

Does not require $\mathcal{X} \subseteq \mathbb{R}^n$. Works with any kind of description variable.

Loss function

$\ell(y^*, H) : \mathbb{R}^2 \rightarrow \mathbb{R}_+$, computes to what extent the inputs match.



$\ell^{0/1}(a, b) = 1_{[sign(a) \neq sign(b)]}$ 0/1 loss ($1_\pi = 1$ iff π **true**, 0 otherwise)
$\ell^{\exp}(a, b) = \exp(-ab)$ exponential loss
$\ell^{\log}(a, b) = \log(1 + \exp(-ab))$ logistic loss
(+ **many** others)

Empirical loss: $\varepsilon_{\mathcal{S}}(H)$

Computes to what extent the labels match between $\mathcal{S}$ and H:

$$\varepsilon_{\boldsymbol{w}_1}(H) \quad \stackrel{\mathrm{def}}{=} \quad \sum_{i=1}^{m} w_{1,i}\ell(y_i^*, H(\boldsymbol{x}_i)) = \mathbf{E}_{(\boldsymbol{x},y^*)\sim \boldsymbol{w}_1}[\ell(y^*, H(\boldsymbol{x}))]$$

True loss: $\varepsilon_{D}(H)$

Computes the real matching between labels, by extending the prediction of H to D:

$$\varepsilon_{D}(H) \quad \stackrel{\mathrm{def}}{=} \quad \mathbf{E}_{(\boldsymbol{x},y^*)\sim D}[\ell(y^*, H(\boldsymbol{x}))]$$

$\ell(.,.) : \mathbb{R}^2 \rightarrow \mathbb{R}_+$ = loss function.

Consider $\ell(.,.) = $ 0/1 loss:

$$\ell^{0/1}(a,b) \stackrel{\mathrm{def}}{=} 1_{[sign(a)\neq sign(b)]} \ ,$$

The losses specialize to the empirical and true **risks**:

$$\begin{aligned}\varepsilon^{0/1}_{\boldsymbol{w}_1}(H) &\stackrel{\mathrm{def}}{=} \mathbf{E}_{(\boldsymbol{x},y^*)\sim \boldsymbol{w}_1}\left[1_{sign(H(\boldsymbol{x}))\neq y^*}\right] \ , \\ \varepsilon^{0/1}_{D}(H) &\stackrel{\mathrm{def}}{=} \mathbf{E}_{(\boldsymbol{x},y^*)\sim D}\left[1_{sign(H(\boldsymbol{x}))\neq y^*}\right]\end{aligned}$$

Remark: if $\mathrm{im}(H) \subseteq \mathbb{R}$, it gives **two** informations:

1. the class, $sign(H)$
2. a **confidence** in the class, $|H|$

Strong Learning

$\mathfrak{A}$ is a Strong learning algorithm for some class of classifiers $\mathcal{H}$ iff

- **Step A** $\mathfrak{A}$ takes as input $0 < \epsilon, \delta \leq 1$
- **Step B** In time poly($1/\epsilon, 1/\delta, n, \ldots$), $\mathfrak{A}$ does the following:
  - **Step B.1** $\mathfrak{A}$ requests a set $\mathcal{S}$ sampled i.i.d. from D
  - **Step B.2** $\mathfrak{A}$ returns $H \in \mathcal{H}$ such that:

$$\mathbf{Pr}_{\mathcal{S}\sim D^m}[\varepsilon_D^{0/1}(H) \leq \epsilon] \quad \geq \quad 1 - \delta$$

Strong because requirements hold for *any* ϵ, δ within bounds.
δ = **confidence** parameter.

Weak Learning

$\cong$ Strong learning, *but* this time ϵ, δ are not user-fixed (Step A):

$$\epsilon \stackrel{\mathrm{def}}{=} 1/2 - \gamma \text{ for some small } \gamma \text{ (\textit{e.g.} cst, } 1/\mathrm{poly}(n)\text{), etc...)}$$
$$\delta \stackrel{\mathrm{def}}{=} \gamma$$

Thus, requirement in B.2 becomes:

$$\mathbf{Pr}_{\mathcal{S}\sim D^m}[\varepsilon_D^{0/1}(H) \leq \underbrace{1/2 - \gamma}_{\text{close to } 1/2}] \geq \underbrace{\gamma}_{\text{close to } 0}$$

Weak because H is only required to carry little “information”:

$$\varepsilon_D(\text{unbiased coin}) = 1/2 \ , \forall D$$

Unbiased coin carries no information about the link Input $\rightarrow$ Output.

The weakening on the confidence is superficial

$$\mathbf{Pr}_{\mathcal{S}\sim D^m}[\underbrace{\varepsilon_D^{0/1}(H) \leq 1/2 - \gamma}_{\text{event } \mathbf{A}}] \geq \gamma$$

Run $T \geq (1/\gamma)\log(1/\delta)$ times Weak learning $\mathfrak{A}$ ($\forall 0 < \delta \leq 1$):

- the probability that event **A** never occurs is $\leq (1-\gamma)^T \leq \exp(-T\gamma) \leq \delta$
- thus, **A** has occurred at least once with probability $\geq 1 - \delta$

..so we can consider that the main difference between Weak and Strong learning relies on the (empirical, true) risks.

To summarize, **Supervised learning**:

1. takes place in **polynomial time**,
2. means building a **classifier** that models a link between **inputs** (observations) and **outputs** (classes)
3. means **controlling** (up to the relevant extent) the **0/1 loss** of this classifier

The third point is the most important for our purpose.

Strong learning

Is strong learning possible ?

Answers depend on $\mathcal{H}$; *many* (early) complexity-theoretic negative results [Kearns and Vazirani(1994)].

Weak learning

Is weak learning possible ?

Some positive answers [Mannor and Meir(2000)]; recent complexity-theoretic negative results [Feldman(2006), Nock and Nielsen(2007b)].

Weak learning $\Rightarrow$ Strong learning (**Boosting**)

Suppose $\mathfrak{A}$ has access to some Weak learning algorithm $\mathfrak{W}$ (Step B.1,5). (How) can $\mathfrak{A}$ meet the requirements of Strong learning ?

An early positive answer [Schapire(1990)].

Learning Strategies

1. Most frequent strategies for Strong, Weak learning, Boosting ?

Principle

Two-step strategy for some $\mathcal{H}$:

- $\mathfrak{A}$ Find a **consistent** classifier H, *i.e.* with $\varepsilon^{0/1}_{\boldsymbol{w}_1}(H) = 0, \forall \mathcal{S}$ (in P-time).
- $\mathcal{H}$ Prove **structural properties**:
  1. on the general $\mathcal{H}$,
  2. or on the subset of $\mathcal{H}$ in which $\mathfrak{A}$ is guaranteed to find H.

Step 2 ensure that we can lift at low (statistical, algorithmic) costs the consistency guarantee on $\mathcal{S}$ to the requirements of Strong learning over D.

Principle

Basically, **computationally tractable** search for a classifier with weak guarantees on the empirical risk.

Typically, two strategies relying on exhaustive search:

1. focused inside a "good" subset of $\mathcal{H}$ if $|\mathcal{H}|$ too large [Mannor and Meir(2000)].
   - \- Algorithmic cost,
   - \+ Theoretically convenient: Weak learning guaranteed.
2. inside some $\mathcal{H}$ of low poly-size. Main experimental approach.
   - \- Sometimes gives up with Weak learning,
   - \+ Empirically convenient: Very fast.

Boosting has to solve a trick:

$$\underbrace{\mathbf{Pr}_{\mathcal{S}\sim D^m}[\varepsilon_D^{0/1}(H) \leq 1/2 - \gamma] \geq \gamma}_{\text{Guaranteed on } \mathfrak{W}\mathfrak{l}}$$

$$\Downarrow$$

$$\underbrace{\mathbf{Pr}_{\mathcal{S}\sim D^m}[\varepsilon_D^{0/1}(H) \leq \epsilon] \geq 1 - \delta}_{\text{Required on } \mathfrak{A}}$$

Recall that boosting the confidence is not a problem. So, how can we "boost" the (empirical, true) risk: $1/2 - \gamma \rightarrow \epsilon$?

(Rough, most popular) Principle, $\forall\mathcal{H}$

Two-step strategy for some $\mathcal{H}$:

$\mathfrak{W}\mathfrak{l}$ Get many (T) classifiers from $\mathfrak{W}\mathfrak{l}$:

$$h_t \quad \leftarrow \quad \mathfrak{W}\mathfrak{l}(\mathcal{S}, \boldsymbol{w}_t) \ , t = 1, 2, ..., T$$

($\mathfrak{W}\mathfrak{l}$ is trained over weight vector that may **differ** from $\boldsymbol{w}_1$)

H Combine the T classifiers to get some $H_T \in \mathcal{H}$.

Very appealing properties:

- for some Boosting algorithms, $\boldsymbol{w}_t$ is repeatedly skewed towards the examples that have been hard to classify so far:

$$\ell^{0/1}(y_i^*, h_t(\boldsymbol{x}_i)) = 1 \quad \Rightarrow \quad w_{t+1,i} > w_{t,i}$$

Boosting Algorithms

1. Most popular algorithms ?
2. Common properties, differences ?

Suppose that $\mathfrak{WI}$ returns $h_t : \mathcal{X} \rightarrow \{-1, +1\}$.

(Discrete) $\mathfrak{A}$daBoost

for $t = 1, 2, ..., T$

1 $h_t \leftarrow \mathfrak{WI}\,(\mathcal{S}, \boldsymbol{w}_t)$;

2 $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t)}{\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t)}$;

3 $\forall (\boldsymbol{x}_i, y_i) \in \mathcal{S}, w_{t+1,i} \leftarrow \frac{w_{t,i} \exp(-y_i \alpha_t h_t(\boldsymbol{x}_i))}{Z_t}$;

($Z_t \stackrel{\mathrm{def}}{=}$ normalization coefficient for $\boldsymbol{w}_t$)

return $H_T = \sum_{t=1}^{T} \alpha_t h_t$;

+ Straightforward to implement, “best off the shelf classifier in the world”;
- restricted outputs for h_t, $\{-1, +1\}$ (generalized in [Friedman et al.(2000)], [Nock and Nielsen(2007a)])

A **fundamental** and simple property of $\mathfrak{A}$daBoost:

1. Compute explicitly the normalization coefficient:

   $Z_t = 2\sqrt{\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t)(1-\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t))}$

2. Unravel the update rule at the end of learning:

   $w_{T+1,i} = w_{1,i}\exp(-y^*_i H_T(\boldsymbol{x}_i))/\prod_t Z_t$

3. Recall that the 0/1 loss is $\leq$ **exponential loss**:

   $\ell^{0/1}(y^*, H_T(\boldsymbol{x})) \leq \ell^{\exp}(y^*, H_T(\boldsymbol{x})) \stackrel{\mathrm{def}}{=} \exp(-y^* H_T(\boldsymbol{x}))$

Sum (2) over $\mathcal{S}$ (this equals 1), multiply both sides by $\prod_t Z_t$, use (1) and get with (3):

$$\varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T) \;\;\leq\;\; 2^T \prod_t \sqrt{\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t)(1-\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t))}$$

Suppose that $\mathfrak{W}$ is a Weak learning algorithm, *i.e.*

$$\varepsilon^{0/1}_{\boldsymbol{w}_t}(h_t) \quad \leq \quad {}^1/_2 - \gamma \quad \text{(whp)} \quad .$$

We get:

$$\varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T) \quad \leq \quad (1-4\gamma^2)^{T/2} \leq \exp(-2\gamma^2 T) \quad .$$

Fixing $T = \Omega((1/\gamma^2)\log m)$ makes the rhs $< 1/m$, *i.e.* (when $\boldsymbol{w}_1 = \boldsymbol{u} \stackrel{\mathrm{def}}{=} ({}^1/_m)\mathbf{1}$):

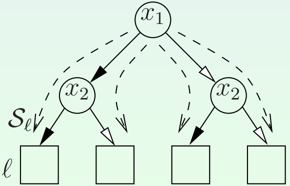
1. H_T is consistent and $\mathfrak{A}$daBoost is P-time;
2. The final LS meets the structural assumptions for Strong learning;

hence, $\mathfrak{A}$daBoost is Strong learning, and thus Boosting.

Property of a decision tree

The empirical risk can be decomposed at the leaves (each leaf sign is the majority class wlog):

$$\begin{aligned}\varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T) &= \sum_{\ell \text{ leaf in } H_T} w_{1,\ell} \min\{w^+_{1,\ell}, 1 - w^+_{1,\ell}\} \\ &= \mathbf{E}_{\ell \sim \boldsymbol{w}_1}\big[\underbrace{\min\{w^+_{1,\ell}, 1 - w^+_{1,\ell}\}}_{\text{minority class \% in } \ell}\big]\end{aligned}$$



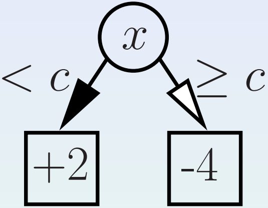
partition of $\mathcal{S}$ in 4 subsets

$\mathcal{S}_\ell$ = subset of $\mathcal{S}$ that reaches leaf ℓ
$w_{1,\ell}$ = total weight of $\mathcal{S}_\ell$ wrt $\boldsymbol{w}_1$
$w^+_{1,\ell}$ = total weight of class $+1$ in $\mathcal{S}_\ell$ wrt $\boldsymbol{w}_1$, divided by $w_{1,\ell}$

e.g. $w^+_{1,\ell} = \sum_{(\boldsymbol{x}_i,+1)\in\mathcal{S}_\ell} w_{1,i} / \sum_{(\boldsymbol{x}_i,y^*_i)\in\mathcal{S}_\ell} w_{1,i}$

Boosting Algorithms > DT (contd)

Suppose that $\mathfrak{WL}$ returns **stumps**, *i.e.* depth-1 DTs.



1. These classifiers are so simple that $\mathfrak{A}$ generally implements $\mathfrak{WL}$ as well...
2. Most popular DT induction algorithms have two stages:
   1. build a large tree ($\mathfrak{T}$DIDT),
   2. prune the tree.

   Here, we (deliberately) reduce them to their first stage.

𝔗op-Down Induction of Decision Trees (𝔗DIDT)

𝔗DIDT (including 𝔚𝔐)

Initialize H_0 to be a single leaf node (=root, $\mathcal{S}_0 = \mathcal{S}$)
For $t = 1, 2, ..., T$

1 pick some leaf ℓ in H_{t-1}.

𝔚𝔐 choose the best stump on $\mathcal{S}_\ell$:

$$h_t^\star = \arg\min_{h_t} \underbrace{\sum_{\ell \text{ leaf in } H_{t-1}\oplus_\ell h_t} w_{1,\ell} \times \phi(w^+_{1,\ell})}_{\text{Splitting criterion} = \varepsilon_{\mathbf{w}_1}(H_{t-1}\oplus h_t, \phi)}$$

2 $H_t \leftarrow H_{t-1} \oplus_\ell h_t^\star$

return $H_T \in$ DT;

$\oplus_\ell h_t$ is the operation that replaces leaf ℓ by h_t, resulting in a DT with one more leaf.

Rewrite the splitting criterion as:

$$\varepsilon_{\boldsymbol{w}_1}(H_T, \phi) = \mathbf{E}_{\ell \sim \boldsymbol{w}_1} \phi(w^{+}_{1,\ell})$$

The empirical risk satisfies $\epsilon^{0/1}_{\boldsymbol{w}_1}(H) = \varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\mathrm{emp}})$ for $\phi_{\mathrm{emp}}(z) \stackrel{\mathrm{def}}{=} \min\{z, 1 - z\}$.

The first $\mathfrak{T}$DIDT scheme, proposed in [Breiman et al.(1984)] proposes a **different** ϕ-criterion:

$$\phi_{\mathrm{CART}}(z) = 2z(1 - z)$$

$\varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\mathrm{CART}})$ is the (expected) **Gini index**

Other popular $\mathfrak{T}$DIDT schemes make the difference only on the choice of ϕ:

$\mathfrak{C}$4.5 [Quinlan(1993)] has

$$\phi_{\mathrm{C4.5}}(z) = -z\log(z) - (1-z)\log(1-z)$$

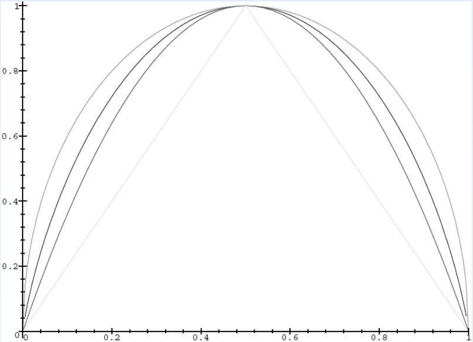
(log base 2)
$\varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\mathrm{C4.5}})$ is the the (expected) **binary entropy**;

$\mathfrak{K}$M [Kearns and Mansour(1999)] has

$$\phi_{\mathrm{KM}}(z) = 2\sqrt{z(1-z)}$$

$\varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\mathrm{KM}})$ is the (expected) **Matsushita error.**

Functions $\phi_{\mathrm{emp}}, \phi_{\mathrm{CART}}, \phi_{\mathrm{C4.5}}, \phi_{\mathrm{KM}}$ have a crucial commonpoint:



Function ϕ is **permissible** iff:

- $\phi : [0, 1] \to [0, 1]$
- $\phi(0) = \phi(1) = 0$
- ϕ sym. wrt $z = 1/2$
- ϕ concave

ϕ **strictly permissible** iff:
permissible and strictly concave

From bottom to top: $2 \times \phi_{\mathrm{emp}}, \phi_{\mathrm{CART}}, \phi_{\mathrm{C4.5}}, \phi_{\mathrm{KM}}$.
All permissible, ϕ_{emp} not strictly permissible.
Normalization for $\phi(1/2) = 1$ not necessary (only for readability).

Fundamental property, easily checked from the last picture:

$$\underbrace{\varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\text{emp}})}_{\text{empirical risk}} \leq \ 1/2 \times \begin{cases} \varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\text{CART}}) \\ \varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\text{C4.5}}) \\ \varepsilon_{\boldsymbol{w}_1}(H_T, \phi_{\text{KM}}) \end{cases}$$

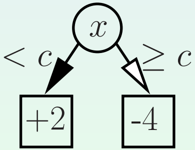
Thus, minimizing any of the right criteria should amount to the minimization of the empirical risk.

$\Rightarrow$ Why so many criteria ?

Consider the following assumption (**W**eak **L**earning **A**ssumption):

WLA $\exists \gamma > 0$ s.t. for any set $\mathcal{S}'$, any distribution $\boldsymbol{w}'$ over $\mathcal{S}'$, there exists a stump h for which:

$$\varepsilon^{0/1}_{\boldsymbol{w}'}(h) \quad \leq \quad 1/2 - \gamma$$



We basically assume that step ((1)+$\mathfrak{W}$l)
is a Weak learning algorithm

Under assumption WLA, the following holds for $\mathfrak{K}$M and its output H_T:

$$T \geq \left(\frac{1}{\epsilon}\right)^{\frac{c}{\gamma^2}} \quad \Rightarrow \quad \varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T) \leq \epsilon$$

c is a constant [Kearns and Mansour(1999), Henry et al.(2007)] (improved c). Fixing $\epsilon < 1/m$ ($\boldsymbol{w}_1 = \boldsymbol{u}$) makes:

1. H_T consistent, and $\mathfrak{K}$M is P-time;
2. The final DT meets the structural assumptions for Strong learning;

hence, $\mathfrak{K}$M is Strong learning, and thus Boosting.

- $\mathfrak{A}$daBoost needs only log calls to $\mathfrak{W}$l compared to $\mathfrak{K}$M, but it is a structural "difficulty" for DT
- the number of calls to $\mathfrak{W}$l is **optimal** for $\mathfrak{K}$M, from both the informational and complexity-theoretic standpoints [Kearns and Mansour(1999), Nock and Nielsen(2004)]

Under assumption WLA, the following holds for $\mathfrak{C}$4.5 and its output H_T:

$$T \geq \left(\frac{1}{\epsilon}\right)^{\frac{c\log(1/\epsilon)}{\gamma^2}} \quad \Rightarrow \quad \varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T) \leq \epsilon$$

Under assumption WLA, the following holds for $\mathfrak{C}$ART and its output H_T:

$$T \geq \left(\frac{1}{\epsilon}\right)^{\frac{c}{\gamma^2\epsilon^2}} \quad \Rightarrow \quad \varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T) \leq \epsilon$$

1. Fix $\epsilon < 1/m$ ($\boldsymbol{w}_1 = \boldsymbol{u}$): the consistency is met, but $\mathfrak{C}$4.5 is QP-time, while $\mathfrak{C}$ART is EXP-time
2. Bounds above are lowerbounds. No tight bound is known, but experimental results seem to confirm the results
3. $\phi = \phi_{\mathrm{emp}}$ would yield the poorest bounds of all (!)

Observation 1 Most DT induction algorithms do not minimize **directly** the empirical risk, but (the expectation of) a *concave* **surrogate** (an upperbound: Gini index, entropy, Mastushita's error)

Observation 2 AdaBoost does not minimize directly the empirical risk, but (the expectation of) a *convex* **surrogate**, the **exponential loss**:

$$\begin{aligned}\varepsilon^{\exp}_{\boldsymbol{w}_1}(H_T) &= \mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[\exp(-y^* H_T(\boldsymbol{x}))] \\ &\geq \varepsilon^{0/1}_{\boldsymbol{w}_1}(H_T)\end{aligned}$$

Observation 3 On DT induction, the **more concave** the permissible function ϕ, the **better** the lowerbounds on T

Observation 4 On DT induction, the direct minimization of the empirical risk yields the **worst possible** lowerbound on T [Nock and Nielsen(2004)]

Supervised learning roughly aims at minimizing empirical risk.

$\boxed{\text{Why focusing on surrogates ?}}$

Numerous well known surrogates:

Concave For DT (and related classes): Gini index, entropy, Matsushita's error

Convex For LS:

$$\begin{aligned}
\varepsilon^{\exp}_{\boldsymbol{w}_1}(H_T) &= \mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[\exp(-y^* H_T(\boldsymbol{x}))] \\
&\quad \text{(Exponential loss: AdaBoost)} \\
\varepsilon^{\log}_{\boldsymbol{w}_1}(H_T) &= \mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[\log(1+\exp(-2y^* H_T(\boldsymbol{x})))] \\
&\quad \text{(Logistic loss)} \\
\varepsilon^{\mathrm{squ}}_{\boldsymbol{w}_1}(H_T) &= \mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[(1-y^* H_T(\boldsymbol{x}))^2] \\
&\quad \text{(Squared loss)...}
\end{aligned}$$

Convex surrogates **have the form** $F(y^* H_T(\boldsymbol{x}))$.

Why focusing on surrogates ?

Explanations so far in this talk:

1. Algorithmic: better convergence properties. Not satisfactory (lack of matching upperbounds).
2. Complexity-theoretic: empirical risk has more local minima (0/1 loss takes on 2 values, thus has less discrimination). Not satisfactory (can be hard for surrogates as well [Nock and Nielsen(2004)]).
3. Others (statistics).

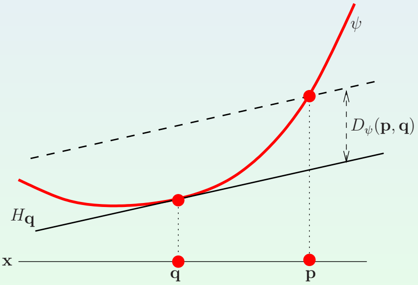
No explanation drills down into the fundamental links between surrogates and classification.

1. Presentation of this class of distortion measures
2. An example of their widespread application in learning

Bregman Divergences (contd)

Let $\psi : \mathcal{X} \rightarrow \mathbb{R}$ strictly convex and differentiable, $\mathcal{X}$ convex. The **Bregman divergence** with **generator** ψ is:

$$D_\psi(\boldsymbol{p}||\boldsymbol{q}) \quad = \quad \psi(\boldsymbol{p}) - \psi(\boldsymbol{q}) - \langle \boldsymbol{p} - \boldsymbol{q}, \nabla_\psi(\boldsymbol{q})\rangle$$



In general, does not satisfy symmetry, triangular inequality.

Squared Euclidean distance

Generator $\psi(\boldsymbol{p}) = \|\boldsymbol{p}\|_2^2$: strictly convex and differentiable over $\mathbb{R}^n$

Divergence

$$\begin{aligned} D_\psi(\boldsymbol{p}\|\boldsymbol{q}) &= \psi(\boldsymbol{p}) - \psi(\boldsymbol{q}) - \langle \boldsymbol{p} - \boldsymbol{q}, \nabla_\psi(\boldsymbol{q}) \rangle \\ &= \|\boldsymbol{p}\|_2^2 - \|\boldsymbol{q}\|_2^2 - \langle \boldsymbol{p} - \boldsymbol{q}, 2\boldsymbol{q} \rangle \\ &= \|\boldsymbol{p} - \boldsymbol{q}\|_2^2 \end{aligned}$$

Generalized I-Divergence

Generator $\psi(\boldsymbol{p}) = \sum_i p_i \log p_i$: strictly convex and differentiable over $\mathbb{R}_+^n$

Divergence

$$D_\psi(\boldsymbol{p}||\boldsymbol{q}) = \sum_i p_i \log\left(\frac{p_i}{q_i}\right) - p_i + q_i$$

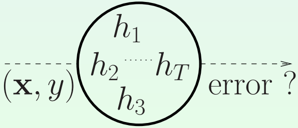
If ψ restricted to the probability simplex, becomes Kullback-Leibler divergence.

The first Supervised learning setting in which they have been explicitly and extensively used.

On-line learning

a We are given a **fixed** set of experts $\{h_i : \mathcal{X} \rightarrow \{-1, +1\}\}_{i=1}^m$, a **stream** of examples. $\boldsymbol{w}_t \in \mathbb{R}_+^m$ is the current set of weights:

1. Receive example $(\boldsymbol{x}_t, y_t^*)$
2. Make prediction $H_m(\boldsymbol{x}) = \sum_i w_{t,i} h_i(\boldsymbol{x})$
3. Incur loss $\ell(y_t^*, H_m(\boldsymbol{x}_t))$
4. Modify the weights: $\boldsymbol{w}_{t+1} \leftarrow f(\boldsymbol{w}_t, \ell(y_t^*, H_m(\boldsymbol{x}_t)))$
5. Go to 1



Objective:
Minimize the number of mistakes

On-line learning is a setting dual to Boosting (reverse the role of the examples and hypotheses in learning). Computation of $\boldsymbol{w}_{t+1}$ involves the aggregation of two Bregman divergences:

$$\boldsymbol{w}_{t+1} \stackrel{\mathrm{def}}{=} \arg\min_{\boldsymbol{w}} \left\{ \underbrace{D_{\psi'}(\boldsymbol{w}||\boldsymbol{w}_t)}_{\text{regularization}} \underbrace{+\eta D_{\psi}\left(\sum_{t=1}^{T} w_t h_t(\boldsymbol{x})|| \underbrace{\nabla_{\psi}^{-1}(y)}_{\in\mathbb{R}}\right)}_{\text{matching loss}} \right\}$$

1. $y \stackrel{\mathrm{def}}{=} (1+y^*)/2 \in \{0,1\}$ is the **Boolean** class.
2. η controls the tradeoff between the two losses.

Axiomatization

1. What is the **true** loss $\ell(y^*, H(\boldsymbol{x}))$ that we (really) want to minimize on each example $(\boldsymbol{x}, y^*)$?
   (we have seen many losses so far: 0/1, convex/concave surrogates, Bregman divergences)
2. Can we find it based on its properties people usually **assume** ?
3. Links with conventional losses ?
4. New losses, families ?

Lifting Classification to **Estimation**

Early "ages" of supervised learning usually preferred the Boolean class:

$$y \stackrel{\mathrm{def}}{=} (1 + y^*)/2 \in \{0, 1\}$$

y is thus a 0/1 estimator for $\mathbf{Pr}[y^* = +1|\boldsymbol{x}]$ (key ingredient for **Bayes rule**).

1. Wlog, assume H able to return an estimator
$$H(\boldsymbol{x}) \rightleftharpoons \hat{\mathbf{Pr}}_H[y^* = +1|\boldsymbol{x}]$$
If $\mathrm{im}(H) \subseteq \mathbb{R}$, $\rightarrow$ done by well-known transfo. (*e.g.* logistic)
If $\mathrm{im}(H) \subseteq [0, 1]$, $\leftarrow$ done by *e.g.* $sign(2H - 1) \in \{-1, +1\}$
2. We end up with the analysis of $\ell(., .)$ with $\mathrm{dom}(\ell) = [0, 1]^2$

Relies on three assumptions in Supervised learning:

1. On the **loss function**
2. On the **best possible rule**
3. On the **cost matrix** for learning

Non Negativity

People assume:

$$\ell(.,.) \quad \geq \quad 0$$

Suppose that **all** examples of $\mathcal{S}$ share the **same** observation $\boldsymbol{x}^\star$. What is the **best constant** prediction for $\boldsymbol{x}^\star$ in average:

$$c = \arg\min_{z\in[0,1]} \mathbf{E}_{(\boldsymbol{x}^\star,y)\sim\boldsymbol{w}_1}[\ell(y,z)] =?$$

Bayes Optimality

People assume that the best prediction rule is Bayes rule:

$$sign(2\mathbf{Pr}[y^* = +1|\boldsymbol{x}] - 1)$$

Thus, the best constant prediction is the best estimator for $\mathbf{Pr}[y^* = +1|\boldsymbol{x}]$, *i.e.*:

$$c = \mathbf{E}_{(\boldsymbol{x}^\star,y)\sim\boldsymbol{w}_1}[y]$$

Fundamental (often implicit) input to supervised learning: the **cost matrix** $L \in \mathbb{R}_+^{2\times 2}$.

		predicted class	
		1	0
true class	1	$\ell(1,1)$	$\ell(1,0)$
	0	$\ell(0,1)$	$\ell(0,0)$

The most general form for the empirical risk is:

$$\varepsilon^{0/1}_{\boldsymbol{w}_1}(H) \stackrel{\mathrm{def}}{=} \mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[\ell(y, 1_{H(\boldsymbol{x})\geq 1/2})]$$

Remark that even right classifications may incur some $\neq 0$ cost.

Symmetric Cost Matrix

People assume* that the **cost matrix** L satisfies the following symmetries:

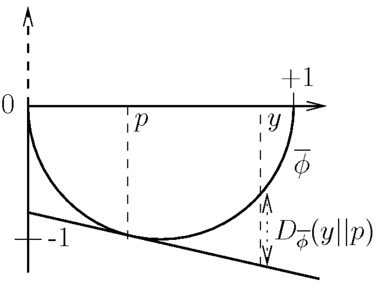
- Diagonal $\ell(1,1) = \ell(0,0)$ ($= 0$: no cost for right classifications)
- Outside $\ell(1,0) = \ell(0,1)$ (same cost for misclassifications)

(this simplifies the empirical risk to the one we have used since the beginning). We thus have:

$$\ell(y,z) \quad = \quad \ell(1-y, 1-z)$$

(*) Holds for domains that have no class-dependent misclassification costs. The others are much less formalized.

Theorem



(here, $\overline{\phi} = -\phi$)

Loss function $\ell(.,.) : [0,1]^2 \rightarrow \mathbb{R}_+$ satisfies assumptions 1, 2, 3

if and only if

$\ell(y, z) = D_{-\phi}(y, z)$
with ϕ **strictly permissible**

(recall DT induction ?)
Strict subclass of Bregman divergences:
Strictly Permissible Bregman Loss Functions (BLF_{SP})

The loss we minimize has thus the general form ($\forall \phi$ strictly permissible):

$$\varepsilon_{\boldsymbol{w}_1}(H) = \varepsilon_{\boldsymbol{w}_1,\overline{\phi}}(H) \stackrel{\mathrm{def}}{=} \underbrace{\mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[D_{\overline{\phi}}(y\|\hat{\mathbf{Pr}}_H[y=1|\boldsymbol{x}])]}_{\text{(expectation of) } \mathrm{BLF}_{\mathrm{SP}}}$$

and we can show:

$$\varepsilon^{0/1}_{\boldsymbol{w}_1}(H) \leq \varepsilon_{\boldsymbol{w}_1,\overline{\phi}}(H)/\phi(1/2)$$

Since $\phi(1/2) \neq 0$, minimizing any $\mathrm{BLF}_{\mathrm{SP}}$ should amount to minimizing the empirical risk as well.

Thus, supervised classification aims at minimizing (the expectation of) some $\mathrm{BLF}_{\mathrm{SP}}$.

Up to a large extent, this means minimizing the empirical risk as well.

What else ?

1. links with surrogates ?
2. minimization algorithms ?

Definition

Suppose ψ strictly convex, differentiable over $\mathbb{X}$. Unique *Convex conjugate* function $\psi^\star$ obtained by the Legendre transformation:

$$\psi^\star(\boldsymbol{q}) = \sup_{\boldsymbol{p}\in\mathbb{X}}\{\langle\boldsymbol{q},\boldsymbol{p}\rangle - F(\boldsymbol{p})\}$$

Solve via $\nabla\psi^\star(\boldsymbol{q}) = \nabla(\langle\boldsymbol{q},\boldsymbol{p}\rangle - \psi(\boldsymbol{p})) = 0$, implying $\boldsymbol{q} = \nabla_\psi(\boldsymbol{p})$, $\boldsymbol{p} = \nabla_\psi^{-1}(\boldsymbol{q})$, and $\psi^\star(\boldsymbol{q}) = \langle\boldsymbol{q},\nabla_\psi^{-1}(\boldsymbol{q})\rangle - \psi(\nabla_\psi^{-1}(\boldsymbol{q}))$.

Dual Bregman divergence

Fundamental link between Bregman divergences:

$$D_\psi(\boldsymbol{p}||\boldsymbol{q}) = D_{\psi^*}(\nabla_\psi(\boldsymbol{q})||\nabla_\psi(\boldsymbol{p}))$$

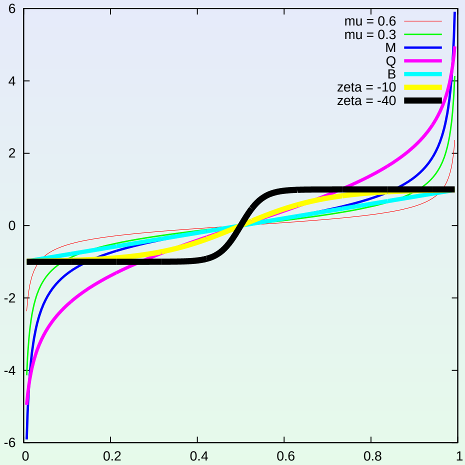
Convex conjugates bring the **link** between $[0,1]$ classification (y) and real-valued classification (y^*). In the BLF$_{\mathrm{SP}}$ $\varepsilon_{\boldsymbol{w}_1,\overline{\phi}}(H)$, we can write:

$$\underbrace{D_{\overline{\phi}}\left(y\|\hat{\mathbf{Pr}}_H[y=1|\boldsymbol{x}]\right)}_{\text{BLF}_{\mathrm{SP}}\text{: }[0,1]\text{ values}} = \underbrace{D_{\overline{\phi}^\star}\left(\nabla_{\overline{\phi}}(\hat{\mathbf{Pr}}_H[y=1|\boldsymbol{x}])\|\nabla_{\overline{\phi}}(y)\right)}_{\text{divergence of real values}}$$

Because ϕ strictly permissible, $\nabla_{\overline{\phi}}$ symmetric wrt $(1/2, 0)$:

1. $\nabla_{\overline{\phi}}(y)$, the $\mathbb{R}$eal class, takes on two **opposite** values
2. Suppose $\mathrm{im}(H) \subseteq \mathbb{R}$. We obtain the transformation rule for the $[0,1]$ values:

$$\hat{\mathbf{Pr}}_H[y=1|\boldsymbol{x}] \stackrel{\mathrm{def}}{=} \nabla_{\overline{\phi}}^{-1}(H(\boldsymbol{x}))$$



"mu" = *upper* regime

$\phi_{M=\mathrm{KM}}(z) = 2\sqrt{z(1-z)}$

$\phi_{Q=\mathrm{C4.5}}(z) = -z\log(z) - (1-z)\log(1-z)$

$\phi_{B=\mathrm{CART}}(z) = 4z(1-z)$

"zeta" = *lower* regime

$\nabla_{\overline{\phi}}(z)$, for various strictly permissible ϕ, depending on its "concavity regime"

Let $\mathrm{im}(H) \subseteq \mathbb{R}$.

Lemma

$$\underbrace{D_{\overline{\phi}}\left(y \| \nabla_{\overline{\phi}}^{-1}(H(\boldsymbol{x}))\right)}_{[0,1] \text{ prediction}} = \underbrace{\overline{\phi}^{\star}(-y^{*}H(\boldsymbol{x}))}_{\mathbb{R}\text{eal prediction}}$$

For any strictly permissible ϕ, $F_{\phi}(z) = \overline{\phi}^{\star}(-z)/\phi(1/2)$ is called a **Permissible Convex Loss** (PCL).

Of course:

$$\varepsilon_{\boldsymbol{w}_1}^{0/1}(H) \leq \underbrace{\mathbf{E}_{(\boldsymbol{x},y^*)\sim \boldsymbol{w}_1}[F_{\phi}(y^{*}H(\boldsymbol{x}))]}_{\text{(expectation of) PCL}}$$

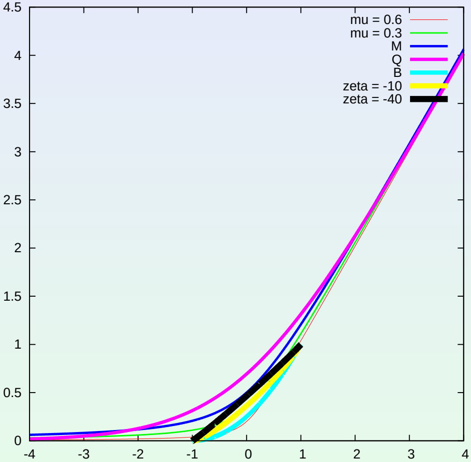
Minimizing any PCL $\Rightarrow$ minimizing the empirical risk.

PCL take on well known special expressions.

$\phi(z)$	$\mathrm{im}H = \mathrm{im}(\nabla_{\overline{\phi}})$	$F_{\phi}(y^* H) = \overline{\phi}^{\star}(-y^* H)/\phi(1/2)$	$\hat{p}_H[y = y_1 \vert \boldsymbol{o}] = \nabla^{-1}_{\overline{\phi}}(H)$
$-z\log z - (1-z)\log(1-z)$	$\mathbb{R}$	$\log(1+\exp(-y^* H))^{*}$	$\frac{\exp(H)}{1+\exp(H)}{}^{*}$
$z(1-z)$	$[-1,1]^{**}$	$(1-y^* H)^{2\,***}$	$(1/2)(1+H)$
$\sqrt{z(1-z)}$	$\mathbb{R}$	$-y^* H + \sqrt{1+(y^* H)^2}$	$\frac{1}{2}\left(1+\frac{H}{\sqrt{1+H^2}}\right)$

- * Logistic loss and logistic transform
- ** Explains why problems when $H \in$ LS.
- *** Squared loss

Many other examples



"mu" = *upper* regime

$\phi_{M=\mathrm{KM}}(z) = 2\sqrt{z(1-z)}$

$\phi_{Q=\mathrm{C4.5}}(z) = -z\log(z) - (1-z)\log(1-z)$

$\phi_{B=\mathrm{CART}}(z) = 4z(1-z)$

"zeta" = *lower* regime

$\overline{\phi}^{\star}(z)$, for various strictly permissible ϕ, depending on its "concavity regime"

We have seen

$$\text{Supervised Learning} \Leftrightarrow \text{min. } \mathrm{BLF}_{\mathrm{SP}} \Leftrightarrow \text{min. PCL}$$

Different standpoints on Supervised classification:

- $\mathrm{BLF}_{\mathrm{SP}}$: $[0, 1]$ classification, H computes **probability estimates**
- PCL: $\mathbb{R}$eal classification, H computes **classes and confidences**

Lemma

$\mathfrak{A}$daBoost's exponential loss is **not** a PCL:

$$\mathbf{E}_{(\boldsymbol{x},y^*)\sim \boldsymbol{w}_1}[\exp(-y^* H_T(\boldsymbol{x}))]$$

Minimization Algorithms

1. Algorithms that minimize some (any) $\mathrm{BLF}_{\mathrm{SP}}$, PCL ?
2. Link with existing algorithms ? New algorithms ?

Minimization Algorithms (contd)

Universal Minimization Algorithm

Let $\mathfrak{A}(\mathcal{S}, \boldsymbol{w}_1, \phi)$ an algorithm that outputs classifiers from set $\mathcal{H}$

1. **If**, for any $\mathcal{S}$, $\boldsymbol{w}_1$, for any strictly permissible ϕ, $\mathfrak{A}$ provably minimizes the corresponding PCL/BLF$_{\mathrm{SP}}$ (see below),
2. **then** $\mathfrak{A}$ is called a **Universal** Minimization Algorithm for $\mathcal{H}$.

$$\underbrace{\mathbf{E}_{(\boldsymbol{x},y^*)\sim\boldsymbol{w}_1}[F_\phi(y^* H(\boldsymbol{x}))]}_{\text{PCL: } \mathrm{im}(H)\subseteq\mathbb{R}} \quad \text{or} \quad \underbrace{\mathbf{E}_{(\boldsymbol{x},y)\sim\boldsymbol{w}_1}[D_{\overline{\phi}}(y\|\hat{\mathbf{Pr}}_H[y=1|\boldsymbol{x}])]}_{\text{BLF}_{\mathrm{SP}}\text{: } \mathrm{im}(H)=[0,1]}$$

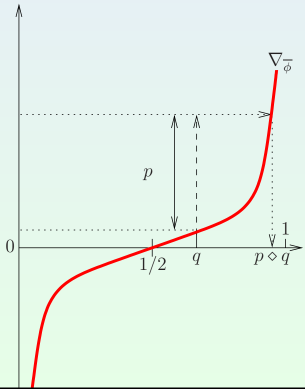
(No P-time complexity requirement)

Any $\mathrm{BLF}_{\mathrm{SP}}$ is convex in its first argument.

Convex conjugates for $\mathrm{BLF}_{\mathrm{SP}}$

Let ϕ strictly permissible. $\forall p \in \mathbb{R}, \forall q \in [0,1]$, the **Legendre dual** $p \diamond q$ of the ordered pair (p, q) is:

$$p \diamond q \stackrel{\mathrm{def}}{=} \arg_{q' \in [0,1]} \sup\{pq' - D_{\overline{\phi}}(q' || q)\} \qquad (= \nabla_{\overline{\phi}}^{-1}(p + \nabla_{\overline{\phi}}(q)))$$



Legendre dual:

1- lifts q to $\mathrm{im}\nabla_{\overline{\phi}}$
2- combines with p
3- maps back to $[0,1]$

A Universal Minimization Algorithm for LS: $\mathfrak{U}$LS.

1. suppose that we already know the set $\{h_1, h_2, ..., h_T\}$, for which $\mathrm{im}(h_t) \subseteq \mathbb{R}$.
2. matrix $M \in \mathbb{R}^{m \times T}$ defined as:

$$m_{it} \stackrel{\mathrm{def}}{=} -y_i^* h_t(\boldsymbol{x}_i) \quad , (\boldsymbol{x}_i, y_i^*) \in \mathcal{S}$$

3. vector notation $(M\boldsymbol{\alpha})_i \stackrel{\mathrm{def}}{=} -y_i^* \underbrace{\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}_i)}_{H(\boldsymbol{x}_i)}, \forall \boldsymbol{\alpha} \in \mathbb{R}^T$
4. Uniform distribution $\boldsymbol{w}_1 \stackrel{\mathrm{def}}{=} \boldsymbol{u} = (1/m)\mathbf{1}$ wlog (duplicate examples)

A None of the h_t has zero empirical risk (otherwise learning not necessary !)

$\mathfrak{U}$LS is mainly a two-step iterative algorithm:

- For $j = 1, 2, ..., J$, do
  1. update the weights over the examples
  2. pick a subset $\mathcal{T}_j \subseteq \{1, 2, ..., T\}$, update the leveraging coefficients of the classifiers $h_t, t \in \mathcal{T}_j$

"$\mathfrak{A}$daBoosting flavor". $\mathfrak{U}$LS specializes in different Boosting schemes:

- classical Boosting framework when $|\mathcal{T}_j| = 1$,
- totally corrective Boosting algorithm when $|\mathcal{T}_j| = \{1, 2, ..., j\}$, etc.

$\mathfrak{U}$LS

Input: $M \in \mathbb{R}^{m \times T}$, strictly permissible ϕ;
Initialize: $\alpha_1 \leftarrow \mathbf{0}$; (leveraging coefficient vector)
Initialize: $\boldsymbol{w}_0 \leftarrow (1/2)\mathbf{1}$; (uniform, **non-unit** weights)
For $j = 1, 2, ..., J$:

1. $\boldsymbol{w}_j \leftarrow (M\boldsymbol{\alpha}_j) \diamond \boldsymbol{w}_0$; (Legendre dual componentwise)
2. Pick $\mathcal{T}_j \subseteq \{1, 2, ..., T\}$ and let $\boldsymbol{\delta}_j \leftarrow \mathbf{0}$;
3. $\forall t \in \mathcal{T}_j$, find $\delta_{j,t}$ such that:
$$\sum_{i=1}^{m} m_{it}((M\boldsymbol{\delta}_j) \diamond \boldsymbol{w}_j)_i = 0$$
4. $\boldsymbol{\alpha}_{j+1} \leftarrow \boldsymbol{\alpha}_j + \boldsymbol{\delta}_j$;

Output: $H(\boldsymbol{x}) \stackrel{\mathrm{def}}{=} \sum_{t=1}^{T} \alpha_{J+1,t} h_t(\boldsymbol{x})$;

Property: (3) has always a solution under **A**.

Theorem

If $\mathcal{T}_j$ is chosen as in classical Boosting, totally corrective Boosting (and others),

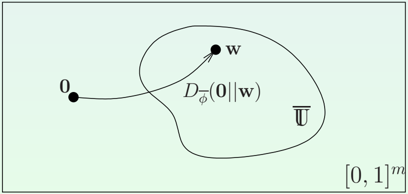
Then $\mathfrak{U}$LS is a Universal Minimization Algorithm.

1. Full Theorem gives the necessary and sufficient conditions on the choice of $\mathcal{T}_j$ for $\mathfrak{U}$LS to remain Universal.
2. $\mathfrak{U}$LS is the largest possible generalization to approaches in [Collins et al.(2002)] (generalizing more implies violating assumptions 1, 2 or 3)
3. Proof unveils the prominent role of “Bregman geometries”

(1): Shift from (P-time) Learning to (Computational) Geometry

$$\underbrace{\min_{\boldsymbol{\alpha}\in\mathbb{R}^T}\sum_{i=1}^{m} F_\phi(y_i^* H(\boldsymbol{x}_i))}_{\text{PCL of the LS}} = \underbrace{\min_{\boldsymbol{w}\in\overline{\mathbb{U}}}\sum_{i=1}^{m} D_{\overline{\phi}}(0||w_i)}_{D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w})}$$

with $\mathbb{U} \stackrel{\mathrm{def}}{=} \{(M\boldsymbol{\alpha})\diamond \boldsymbol{w}_0 : \boldsymbol{\alpha}\in\mathbb{R}^T\}$ (recall $\boldsymbol{w}_0 \stackrel{\mathrm{def}}{=} (1/2)\mathbf{1}$)

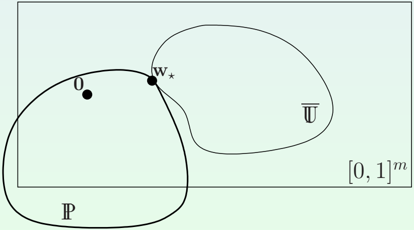


$\mathbf{0} \notin \overline{\mathbb{U}}$ under **A**

(2): Existence of a particular point in $\overline{\mathbb{U}}$

$$\forall \boldsymbol{w}_\star \in \mathbb{R}^m, \boldsymbol{w}_\star \in \mathbb{P} \cap \overline{\mathbb{U}} \quad \Leftrightarrow \boldsymbol{w}_\star = \arg\min_{\boldsymbol{w} \in \overline{\mathbb{U}}} D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w})$$

with $\mathbb{P} \stackrel{\mathrm{def}}{=} \{\boldsymbol{z} \in \mathbb{R}^m : M\boldsymbol{z} = \mathbf{0}\} = \mathrm{Ker}M$
(recall $\mathbb{U} \stackrel{\mathrm{def}}{=} \{(M\boldsymbol{\alpha}) \diamond \boldsymbol{w}_0 : \boldsymbol{\alpha} \in \mathbb{R}^T\}$ and $\boldsymbol{w}_0 \stackrel{\mathrm{def}}{=} (1/2)\mathbf{1}$)



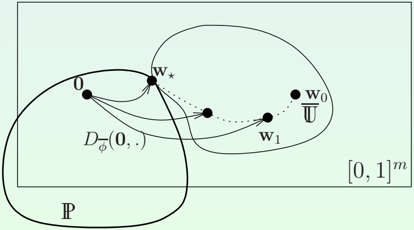
($\boldsymbol{w}_\star$ is unique)

Objective: find $\boldsymbol{w}_\star$

(3): $\mathfrak{U}$LS is a constrained minimization algorithm

Recall that $\mathfrak{U}$LS builds a sequence $\boldsymbol{w}_0, \boldsymbol{w}_1, ..., \boldsymbol{w}_J \in \overline{\mathbb{U}}$.
Let an **auxiliary function** $u : [0,1]^m \times [0,1]^m \rightarrow \mathbb{R}$ for algorithm $\mathfrak{U}$LS be a function that would satisfy:

$$
\begin{aligned}
D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w}_{j+1}) - D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w}_j) \leq u(\boldsymbol{w}_{j+1}, \boldsymbol{w}_j) &\leq 0 & \text{(i)}\\
u(\boldsymbol{w}_{j+1}, \boldsymbol{w}_j) = 0 &\Rightarrow M\boldsymbol{w}_{j+1} = \mathbf{0} & \text{(ii)}
\end{aligned}
$$



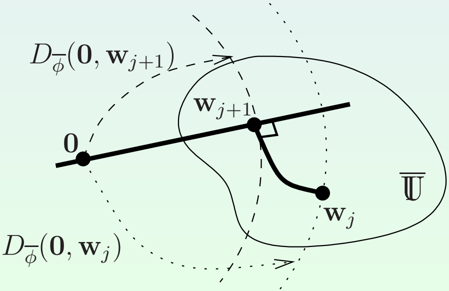
If u exists,
(i) $\mathfrak{U}$LS provably minimizes $D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w})$, and in $\overline{\mathbb{U}}$.
(ii) upon convergence, $\mathfrak{U}$LS ends up with some $\boldsymbol{w}_J \in \mathbb{P}$
Hence, $\boldsymbol{w}_J \in \overline{\mathbb{U}} \cap \mathbb{P}$
Hence, $\boldsymbol{w}_J = \boldsymbol{w}_\star = \arg\min_{\boldsymbol{w}} D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w})$

(4): The auxiliary function for 𝔘LS

The computation of $\boldsymbol{w}_j$ and δ_j in 𝔘LS yields

$$D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w}_{j+1}) - D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w}_j) = \underbrace{-D_{\overline{\phi}}(\boldsymbol{w}_{j+1}||\boldsymbol{w}_j)}_{u(\boldsymbol{w}_{j+1},\boldsymbol{w}_j)}$$

$u(\boldsymbol{w}_{j+1}, \boldsymbol{w}_j) \leq 0$, equality iff $\boldsymbol{w}_{j+1} = \boldsymbol{w}_j$ (prop of Bregman div.)



Generalized Pythagoras Theorem

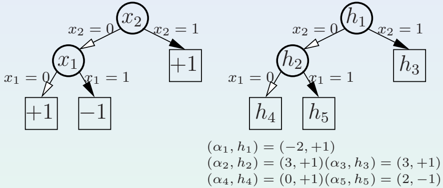
$$D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w}_j) = D_{\overline{\phi}}(\mathbf{0}||\boldsymbol{w}_{j+1}) + D_{\overline{\phi}}(\boldsymbol{w}_{j+1}||\boldsymbol{w}_j)$$

Summary:

1. 𝔘LS is a Universal Minimization Algorithm
2. Uses geometric properties on the weight vectors $\boldsymbol{w}$ to converge
3. Under some Weak Learning Assumption about the h_t, (loose) convergence rates

What about DT ? Any 𝔘niversal minimization algorithm for DT ?

We do not have to think everything from scratch: use **Linearized** Decision Trees [Henry et al.(2007)]



In a LDT,

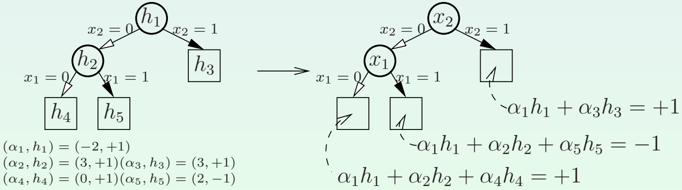
1. reals on every node (not just leaves)
2. sum the reals over a path to decide the class
3. each path is a **constant** LS

Twin DT

From any LDT, find the **twin** DT: for each path root — leaf,

1. computes the constant LS,
2. put the value at the leaf

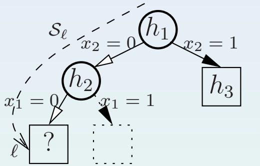
At the end, remove in all internal nodes any couple (α, h). We obtain a DT equivalent to the LDT.



Minimization Algorithms > $\mathfrak{U}$DT

$\mathfrak{U}$DT recycles $\mathfrak{U}$LS on a strategy that minimizes the corresponding PCL

- To fit the internal couples (α, h), use $\mathfrak{U}$LS$(\mathcal{S}_\ell, \boldsymbol{w}_{1,\ell}, \phi)$



$(\alpha_1, h_1) = (-2, +1)$
$(\alpha_2, h_2) = (3, +1)(\alpha_3, h_3) = (3, +1)$
$(\alpha_4, h_4) = (?, ?)$

- To find the splits, further minimize the global PCL over the choice of splits:

  find the split replacing leaf ℓ which minimizes $\mathbf{E}_{(\boldsymbol{x}, y^*) \sim \boldsymbol{w}_1}[F_\phi(y^* H(\boldsymbol{x}))]$

Lemma

$\mathfrak{U}$DT is a Universal Minimization Algorithm

Is it known ?...

Theorem

Suppose we replace the LDT output by its twin DT. $\mathfrak{U}$DT($\mathcal{S}$, $\boldsymbol{w}_1$, ϕ) simplifies **exactly** to the general $\mathfrak{T}$DIDT scheme that minimizes $\varepsilon_{\boldsymbol{w}_1}(H_T, \phi)$ over the DT

Consequences (examples):

1. binds the most popular induction schemes for LS and DT as the **same** (master) algorithm, that uses the same geometric properties
2. $\mathfrak{A}$daBoost and $\mathfrak{K}$M are the same algorithm
3. the exact optimization of the logistic loss [Friedman et al.(2000)] is the **same** algorithm as $\mathfrak{C}$4.5

What about AdaBoost ?

Recall that the exponential loss of AdaBoost is not permissible... but:

Theorem

(a **very** slight) modification of AdaBoost is a Universal Minimization Algorithm

1. A single loss helps to minimize all !
2. Not surprising: minimizing any $\mathrm{BLF}_{\mathrm{SP}}$ amounts to a Maximum Likelihood estimation to fit Bernoulli or Laplace priors.

Perspectives

1. **Transfer positive results**: we can use this master algorithm to obtain (new) formal Boosting algorithm for well known other classes $\mathcal{H}$:
   - The algorithm fits local linear separators on a particular **decision graph**
   - We can combine the same algorithm in a **recursive** fashion:
     1. we obtain $\mathfrak{A}$,
     2. we obtain $\mathfrak{W}$ **with the same algorithm**,
     3. and we can drill down even further...

Perspectives (contd)

2. **Transfer negative results**: we can translate back and forth bounds to get hint on the hardness of learning for particular $\mathcal{H}$
   - Example: bounds of [Kearns and Mansour(1999)] on DT can be translated to LS
   - We get explicit bounds for the fact that the exact minimization of the logistic loss [Friedman et al.(2000)] may not be as efficient as $\mathfrak{A}$daBoost
   - Optimizing the squared loss would be even less efficient
   - optimizing the empirical would be the less efficient of all criteria (!)

Perspectives (contd)

3. **No-Free lunch Theorems**: any algorithm has hard problems
   - Find a good parameterization for ϕ: can we learn it while learning the data (self improving algorithms) ?

Thank you for your attention

Acknowledgments:

- work done in collaboration with
    1. Frank Nielsen @ Sony CSL Tokyo
    2. Claudia Henry (ANR/MESR PhD student)
- support from ANR, programme “Jeunes Chercheurs” JC 9009

Thank you for your attention

For more information:

1. see paper [Henry et al.(2007)] and longer version:
   Boosting does not get Lost in Translation
   (Nock, Henry, Nielsen), 45pp, submitted
2. see also:
   On Permissible Surrogates for Classification
   (Nock, Nielsen), 52pp, submitted

(available upon request)

Bibliography

Breiman, L., Freidman, J. H., Olshen, R. A., Stone, C. J., 1984. Classification and regression trees. Wadsworth.

Collins, M., Schapire, R., Singer, Y., 2002. Logistic regression, adaboost and Bregman distances. Machine Learning , 253–285.

Feldman, V., 2006. Optimal hardness results for maximizing agreements with monomials. In: Proc. of the 21 *st* IEEE International Conference on Computational Complexity.

Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive Logistic Regression : a Statistical View of Boosting. Ann. of Stat. 28, 337–374.

Henry, C., Nock, R., Nielsen, F., 2007. Real boosting *a la Carte* with an application to boosting Oblique Decision Trees. In: Proc. of the 21 *st* International Joint Conference on Artificial Intelligence.

Kearns, M., Mansour, Y., 1999. On the boosting ability of top-down decision tree learning algorithms. Journal of Computer and System Sciences 58, 109–128.

Kearns, M. J., Vazirani, U. V., 1994. An Introduction to Computational Learning Theory. M.I.T. Press.

Mannor, S., Meir, R., 2000. Weak learners and improved rates of convergence in boosting. In: Advances in Neural Information Processing Systems 13.

Nock, R., Nielsen, F., 2004. On Domain-Partitioning Induction Criteria: Worst-case Bounds for the Worst-case Based. Theoretical Computer Science 321, 371–382.

Nock, R., Nielsen, F., 2007a. A $\mathbb{R}$eal Generalization of discrete AdaBoost. Artificial Intelligence 171, 25–41.

Nock, R., Nielsen, F., 2007b. Self-Improved gaps Almost Everywhere for the Agnostic Approximation of Monomials. Theoretical Computer Science 377, 139–150.

Quinlan, J. R., 1993. C4.5 : programs for machine learning. Morgan Kaufmann.