Binarsity: a penalization for one-hot encoded features in linear supervised learning

Mokhtar Z. Alaya

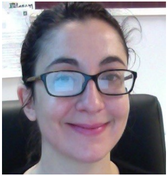
Université Paris Nanterre

AgroParisTech Seminar, November 2018

Joint work with ...

Simon Bussy
Laboratoire de Probabilités Statistique et Modélisation
Université Pierre et Marie Curie

Stéphane Gaïffas
Laboratoire de Probabilités Statistique et Modélisation
Université Paris Diderot



Agathe Guilloux
Laboratoire de Mathématiques et Modélisation d'Évry
Université d'Évry Val d'Essonne

Outline

1. Weighted total-variation penalization
2. Binarsity
   - Features binarization
   - Binarsity penalization
3. Generalized linear models + binarsity
4. Numerical experiements

Supervised learning: framework

Setting

- Data $D_n = \{(X_i, Y_i) : i = 1, \dots, n\}$ supposed to be i.i.d.
- $X_i \in \mathcal{X} = \mathbb{R}^p$, $Y_i \in \mathcal{Y}$ for $i = 1, \dots, n$. The X_i are called **features** and the Y_i are called **labels**.
- The labels are scalar numbers. We assume that $\mathcal{Y} \subset \mathbb{R}$. $\mathcal{Y} = \mathbb{R}$ for regression, $\mathcal{Y} = \{-1, +1\}$ for binary classification.

High-dimension

- p is larger.

Big data

- n is larger.

Goal

- Based on (x_i, y_i), learn a function that predicts y based on a new x (generalization property).

Supervised learning: empirical risk + penalization

Minimize with respect to $h : \mathbb{R}^p \to \mathbb{R}$

$$R_n(h) + \lambda \text{pen}(h)$$

where

- $$R_n(h) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, h(x_i))$$

  is an **empirical risk**, where ℓ is a **loss** function.
- pen is a penalization function, that encodes a prior assumption on h.
- $\lambda > 0$ is a **tuning parameter**, that balances good-of-fitness and penalization.
- **Simplification**: choose a **linear** function f:

  $$h(x) = x^\top \theta = \sum_{j=1}^{p} x_j \theta_j,$$

  for a parameter $\theta \in \mathbb{R}^p$ to be trained.

- We end up with:

$$\hat{\theta} \in \underset{\theta \in \mathbb{R}^p}{\text{argmin}}\{R_n(\theta) + \lambda \text{pen}(\theta)\},$$

where

$$R_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, x_i^\top \theta)$$

and pen(θ) is a penalization on θ.

- Choice of penalization !

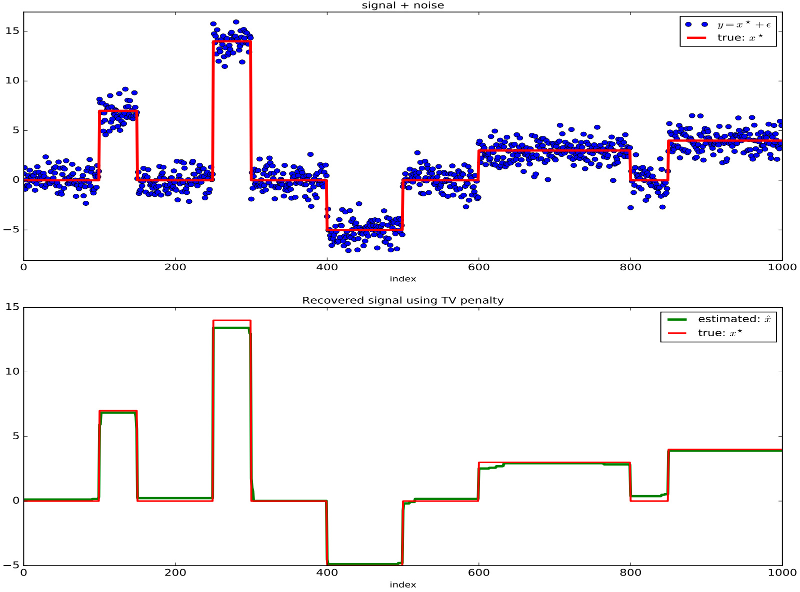
- ℓ_0-quasi-norm: $\text{pen}(\theta) = \|\theta\|_0 = \#\{j : \theta_j \neq 0\}$.
- Lasso (ℓ_1-norm): $\text{pen}(\theta) = \|\theta\|_1 = \sum_{j=1}^{p} |\theta_j|$ [Tibshirani (1996)].
- Elastic-Net ($(\ell_1 + \ell_2^2)$-norm): $\text{pen}(\theta) = \|\theta\|_1 + \|\theta\|_2^2$ [Zou and Hastie (2005)].
- Fused Lasso (ℓ_1 + TV): $\text{pen}(\theta) = \|\theta\|_1 + \|\theta\|_{\text{TV}}$ [Tibshirani et al. (2005)] where $\|\cdot\|_{\text{TV}}$ is the (discrete) total-variation penalization (TV) defined as

$$\|\theta\|_{\text{TV}} = \sum_{j=2}^{p} |\theta_j - \theta_{j-1}|, \text{ for all } \theta \in \mathbb{R}^p.$$

Motivations for using TV

- Appropriate for multiple change-points estimation.
  $\longrightarrow$ Partitioning a nonstationary signal into several contiguous stationary segments of variable duration [Harchaoui and Lévy-Leduc (2010), Alaya et al. (2015)].
- Widely used in sparse signal processing and imaging (2D) [Chambolle et al. (2010)].
- Enforces sparsity in the discrete gradient, which is desirable for applications with features ordered in some meaningful way [Tibshirani et al. (2005)].

Toy example: recovery of piecewise constant signal using TV



Weighted TV

- For a chosen positive vector of weights $\hat{\omega}$, we define the weighted TV by

$$\|\theta\|_{\mathrm{TV},\hat{\omega}} = \sum_{j=2}^{p} \hat{\omega}_j |\theta_j - \theta_{j-1}|.$$

- If $\hat{\omega} \equiv 1$, then we get the simple (unweighted) TV by

$$\|\theta\|_{\mathrm{TV},1} = \|\theta\|_{\mathrm{TV}} = \sum_{j=2}^{p} |\theta_j - \theta_{j-1}|.$$

Proximal operator of weighted TV penalization

- We are interested in computing a solution

$$\hat{\theta} = \underset{\theta}{\text{argmin}}\{f(\theta) + g(\theta)\},$$

where f is smooth and g is simple (prox-calculable).

- The proximal operator prox_g of a proper, lower semi-continuous, convex function $h : \mathbb{R}^n \to (-\infty, \infty]$, is defined as

$$\text{prox}_{\lambda g}(y) = \underset{\theta \in \mathbb{R}^n}{\text{argmin}} \Big\{ \frac{1}{2}\|y - \theta\|_2^2 + \lambda g(\theta) \Big\}, \text{ for all } y \in \mathbb{R}^n.$$

- Proximal gradient descent (PGD) algorithm is based on

$$\theta^{(t+1)} = \text{prox}_{\eta_t g}\big(\theta^{(t)} - \eta_t \nabla f(\theta^{(t)})\big).$$

[ISTA Daubechies et al. (2004), FISTA Beck and Teboulle (2009)]

Proximal operator of weighted TV penalization

- We have

$$\hat{\theta} = \text{prox}_{\|\cdot\|_{\text{TV},\hat{\omega}}}(y) = \underset{\theta \in \mathbb{R}^n}{\text{argmin}} \Big\{ \frac{1}{2}\|y - \theta\|_2^2 + \|\theta\|_{\text{TV},\hat{\omega}} \Big\}.$$

- Modification of Condat's algorithm [Condat (2013)].
- If we have a feasible dual variable $\hat{u}$, we can compute the primal solution $\hat{\theta}$, by Fenchel duality.
- The Karush-Kuhn-Tucker (KKT) optimality conditions [Boyd and Vandenberghe (2004)] characterize the unique solutions $\hat{\theta}$ and $\hat{u}$.

Algorithm 1: $\hat{\theta} = \text{prox}_{\|\cdot\|_{TV,\hat{\omega}}}(y)$ [Alaya et al. (2015)]

1. **set** $k = k_0 = k_- = k_+ \leftarrow 1$; $\theta_{\min} \leftarrow y_1 - \hat{\omega}_2$; $\theta_{\max} \leftarrow y_1 + \hat{\omega}_2$; $u_{\min} \leftarrow \hat{\omega}_2$; $u_{\max} \leftarrow -\hat{\omega}_2$;
2. **if** $k = n$ **then**
   - $\hat{\theta}_n \leftarrow \theta_{\min} + u_{\min}$;
3. **if** $y_{k+1} + u_{\min} < \theta_{\min} - \hat{\omega}_{k+2}$ **then** `/* negative jump */`
   - $\hat{\theta}_{k_0} = \cdots = \hat{\theta}_{k_-} \leftarrow \theta_{\min}$; $k = k_0 = k_- = k_+ \leftarrow k_- + 1$;
   - $\theta_{\min} \leftarrow y_k - \hat{\omega}_{k+1} + \hat{\omega}_k$; $\theta_{\max} \leftarrow y_k + \hat{\omega}_{k+1} + \hat{\omega}_k$; $u_{\min} \leftarrow \hat{\omega}_{k+1}$; $u_{\max} \leftarrow -\hat{\omega}_{k+1}$;
4. **else if** $y_{k+1} + u_{\max} > \theta_{\max} + \hat{\omega}_{k+2}$ **then** `/* positive jump */`
   - $\hat{\theta}_{k_0} = \ldots = \hat{\theta}_{k_+} \leftarrow \theta_{\max}$; $k = k_0 = k_- = k_+ \leftarrow k_+ + 1$;
   - $\theta_{\min} \leftarrow y_k - \hat{\omega}_{k+1} - \hat{\omega}_k$; $\theta_{\max} \leftarrow y_k + \hat{\omega}_{k+1} - \hat{\omega}_k$; $u_{\min} \leftarrow \hat{\omega}_{k+1}$; $u_{\max} \leftarrow -\hat{\omega}_{k+1}$;
5. **else** `/* no jump */`
   - **set** $k \leftarrow k + 1$; $u_{\min} \leftarrow y_k + \hat{\omega}_{k+1} - \theta_{\min}$; $u_{\max} \leftarrow y_k - \hat{\omega}_{k+1} - \theta_{\max}$;
   - **if** $u_{\min} \geq \hat{\omega}_{k+1}$ **then**
     - $\theta_{\min} \leftarrow \theta_{\min} + \frac{u_{\min} - \hat{\omega}_{k+1}}{k - k_0 + 1}$; $u_{\min} \leftarrow \hat{\omega}_{k+1}$; $k_- \leftarrow k$;
   - **if** $u_{\max} \leq -\hat{\omega}_{k+1}$ **then**
     - $\theta_{\max} \leftarrow \theta_{\max} + \frac{u_{\max} + \hat{\omega}_{k+1}}{k - k_0 + 1}$; $u_{\max} \leftarrow -\hat{\omega}_{k+1}$; $k_+ \leftarrow k$;
6. **if** $k < n$ **then**
   - **go to 3.**;
7. **if** $u_{\min} < 0$ **then**
   - $\hat{\theta}_{k_0} = \cdots = \hat{\theta}_{k_-} \leftarrow \theta_{\min}$; $k = k_0 = k_- \leftarrow k_- + 1$; $\theta_{\min} \leftarrow y_k - \hat{\omega}_{k+1} + \hat{\omega}_k$;
   - $u_{\min} \leftarrow \hat{\omega}_{k+1}$; $u_{\max} \leftarrow y_k + \hat{\omega}_k - v_{\max}$; **go to 2.**;
8. **else if** $u_{\max} > 0$ **then**
   - $\hat{\theta}_{k_0} = \cdots = \hat{\theta}_{k_+} \leftarrow \theta_{\max}$; $k = k_0 = k_+ \leftarrow k_+ + 1$; $\theta_{\max} \leftarrow y_k + \hat{\omega}_{k+1} - \hat{\omega}_k$;
   - $u_{\max} \leftarrow -\hat{\omega}_{k+1}$; $u_{\min} \leftarrow y_k - \hat{\omega}_k - u_{\min}$; **go to 2.**;
9. **else**
   - $\hat{\theta}_{k_0} = \cdots = \hat{\theta}_n \leftarrow \theta_{\min} + \frac{u_{\min}}{k - k_0 + 1}$;

Features binarization

- Supervised training dataset $(x_i, y_i)_{i=1,\ldots,n}$ containing features $x_i = (x_{i,1}, \ldots, x_{i,p})^\top \in \mathbb{R}^p$ and labels $y_i \in \mathcal{Y} \subset \mathbb{R}$, that are i.i.d.
- We denote $\boldsymbol{X} = [x_{i,j}]_{1 \leq i \leq n; 1 \leq j \leq p}$ the $n \times p$ features matrix.
- Let $\boldsymbol{X}_{\bullet,j}$ be the j-th feature column of $\boldsymbol{X}$.
- The j-th column $\boldsymbol{X}_{\bullet,j}$ is replaced by a number $d_j \geq 2$ of columns $\boldsymbol{X}^B_{\bullet,j,1}, \ldots, \boldsymbol{X}^B_{\bullet,j,d_j}$ containing only zeros and ones.
- The binarized matrix $\boldsymbol{X}^B$ is a matrix with an extended number $d = \sum_{j=1}^p d_j > p$ of columns (only binary).

Features binarization: setup

- If $X_{\bullet,j}$ takes values (modalities) in the set $\{1, \ldots, M_j\}$ with cardinality M_j, we take $d_j = M_j$, and use a one-hot coding of each modality by defining

$$x^B_{i,j,k} = \begin{cases} 1, & \text{if } x_{i,j} = k, \\ 0, & \text{otherwise,} \end{cases}$$

- If $X_{\bullet,j}$ is quantitative, then d_j we consider a partition of intervals $I_{j,1}, \ldots, I_{j,d_j}$ for the range of values of $\boldsymbol{X}_{\bullet,j}$ and define

$$x^B_{i,j,k} = \begin{cases} 1, & \text{if } x_{i,j} \in I_{j,k}, \\ 0, & \text{otherwise,} \end{cases}$$

- The i-th raw of $\boldsymbol{X}^B$ is written

$$x^B_i = (x^B_{i,1,1}, \ldots, x^B_{i,1,d_1}, x^B_{i,2,1}, \ldots, x^B_{i,2,d_2}, \ldots, x^B_{i,p,1}, \ldots, x^B_{i,p,d_p})^\top \in \mathbb{R}^d$$

Features binarization: setup

- A natural choice of intervals is given by the quantiles, namely we can typically choose $I_{j,k} = \big(q_j(\frac{k-1}{d_j}), q_j(\frac{k}{d_j})\big]$ for $k = 1, \ldots, d_j$, where $q_j(\alpha)$ denotes a quantile of order $\alpha \in [0,1]$ for $\boldsymbol{X}_{\bullet,j}$.
- ▸ Example of features binarization using tick library Python library "tick" [Bacry et al. (2018)]
- To each binarized feature $\boldsymbol{X}^B_{\bullet,j,k}$ corresponds a parameter $\theta_{j,k}$.
- The parameters associated to the binarization of the j-th feature is denoted $\theta_{j,\bullet} = (\theta_{j,1} \cdots \theta_{j,d_j})^\top$.
- The full parameters vector of size $d = \sum_{j=1}^p d_j$, is simply

$$\begin{aligned}\theta &= (\theta_{1,\bullet}^\top \cdots \theta_{p,\bullet}^\top)^\top \\ &= \big(\theta_{1,1} \cdots \theta_{1,d_1}\, \theta_{2,1} \cdots \theta_{2,d_2} \cdots \theta_{p,1} \cdots \theta_{p,d_p}\big)^\top \in \mathbb{R}^d.\end{aligned}$$

Features binarization: some issues

- (P1) The one-hot-encodings satisfy $\sum_{k=1}^{d_j} \boldsymbol{X}_{i,j,k} = 1$ for all j, meaning that the columns of each block sum to $\mathbf{1}_n$.
  $\rightarrow$ $\boldsymbol{X}^B$ is not of full rank by construction.
- (P2) Over-parametrization: increasing the number of d_j for binarization of each row feature j is not an easy task leads to overfitting.
- (P3) Some of the raw features $\boldsymbol{X}_{\bullet,j}$ might not be relevant for the prediction task, so we want to select raw features from their one-hot encodings,
  $\rightarrow$ **block-sparsity** in θ.

- To deal with (P1), we impose a linear constraint in each block. In our penalization term, we impose a **sum-to-zero-constraint**, that is

$$\sum_{k=1}^{d_j} \theta_{j,k} = 0 \text{ for all } j = 1, \dots, p.$$

- To tackle (P2), we keep the number of different values taken by $\theta_{j,\bullet}$ to minimal level by using a within block weighted total-variation penalization

$$\sum_{j=1}^{p} \|\theta_{j,\bullet}\|_{\mathrm{TV},\hat{\omega}_{j,\bullet}} = \sum_{k=2}^{d_j} \hat{\omega}_{j,k} |\theta_{j,k} - \theta_{j,k-1}|$$

- We therefore introduce the following new penalization called *binarsity*

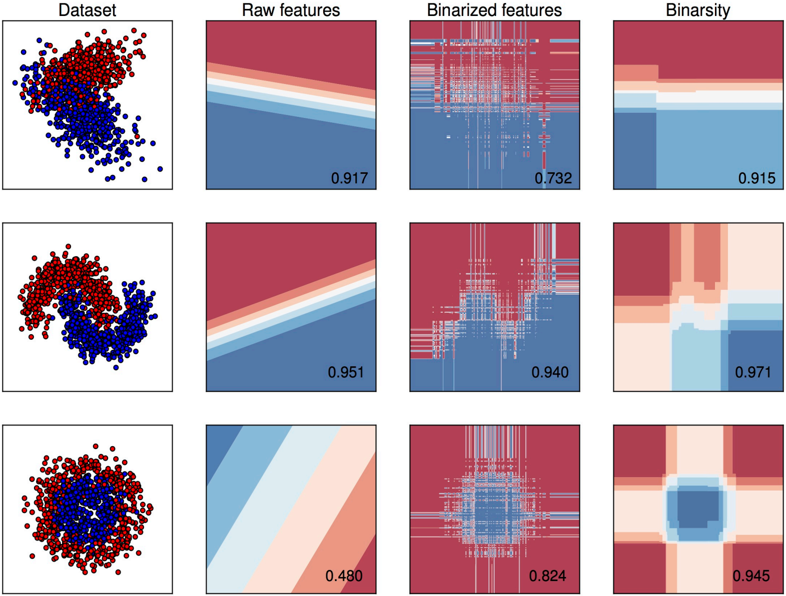
$$\mathrm{bina}(\theta) = \sum_{j=1}^{p} \Big(\sum_{k=2}^{d_j} \hat{\omega}_{j,k} |\theta_{j,k} - \theta_{j,k-1}| + \delta_1(\theta_{j,\bullet}) \Big),$$

where the indicator function

$$\delta_1(u) = \begin{cases} 0 & \text{if} \quad \mathbf{1}^\top u = 0, \\ \infty & \text{otherwise.} \end{cases}$$

- If a raw feature j is statistically not relevant for predicting the labels, then the full block $\theta_{j,\bullet}$ should be zero.
- If a raw feature j is relevant, then the number of different values for the coefficients of $\theta_{j,\bullet}$ should be kept as small as possible, in order to balance bias and variance.

Toy example ($n = 1000$, $p = 2$, $d_1 = d_2 = 100$)



Generalized linear models

- The conditional distribution of Y_i given $X_i = x_i$ is assumed to be from one parameter exponential family

$$y|x \mapsto f^0(y|x) = \exp\Big(\frac{y m^0(x) - b(m^0(x))}{\phi} + c(y,\phi)\Big),$$

- The functions $b(\cdot)$ and $c(\cdot)$ are known, while the natural parameter function $m^0(x)$ is *unknown*.
- We have

$$m^0(x) = g(\mathbb{E}[Y_i|X_i = x_i]), \quad \text{where } b' = g^{-1}.$$

- Logistic and probit regression for binary data or multinomial regression for categorical data, Poisson regression for count data, etc ...

Generalized linear models + binarsity

- We consider the empirical risk

$$R_n(m_\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, m_\theta(x_i)\big),$$

where

$$m_\theta(x_i) = \theta^\top x_i^B.$$

- ℓ is the generalized linear model loss function and is given by

$$\ell\big(y, y'\big) = -yy' + b(y').$$

- Our estimator of m^0 is given by $\hat{m} = m_{\hat{\theta}}$, where $\hat{\theta}$ is the solution of the penalized log-likelihood problem

$$\hat{\theta} \in \underset{\theta \in \mathbb{R}^d}{\text{argmin}} \big\{ R_n(m_\theta) + \text{bina}(\theta) \big\}.$$

- To evaluate the quality of the estimation, we shall use the *excess risk* of $\hat{m}$,

$$R(\hat{m}) - R(m_0) = \mathbb{E}_{\mathscr{L}(Y|X)}[R_n(\hat{m}) - R_n(m_0)].$$

- We consider the following data-driven weighted version of Binarsity given by

$$\hat{\omega}_{j,k} = \mathcal{O}\Bigg(\sqrt{\frac{\log d}{n}\hat{\pi}_{j,k}}\Bigg),$$

where

$$\hat{\pi}_{j,k} = \frac{\#\Big(\Big\{i = 1, \ldots, n : x_{i,j} \in \big(q_j\big(\frac{k}{d_j}\big), q_j(1)\big]\Big\}\Big)}{n}.$$

$\hat{\pi}_{j,k}$ corresponds to the proportion of 1s in the sub-matrix obtained by deleting the first k columns in the j-th binarized block matrix.

Assumption

Assume that b is three times continuously differentiable, and that there exist constants $C_n > 0$, *and* $0 < L_n \leq U_n$ *such that* $C_n = \max_{i=1,\dots,n} |m^0(x_i)| < \infty$ *and* $L_n \leq \max_{i=1,\dots,n} b''\big(m^0(x_i)\big) \leq U_n$.

For all $\theta \in \mathbb{R}^d$, let $J(\theta) = \big[J_1(\theta), \dots, J_p(\theta)\big]$ be the concatenation of the support sets relative to the total-variation penalization, that is

$$J_j(\theta) = \big\{k : \theta_{j,k} \neq \theta_{j,k-1}, \text{ for } k = 2, \dots, d_j\big\}.$$

Assumption

Let $K = [K_1, \dots, K_p]$ *be a concatenation of index sets such that* $\sum_{j=1}^p |K_j| \leq J^\star$. *Assume*

$$\kappa(K) \in \inf_{u \in \mathscr{C}_{\mathrm{TV},\hat\omega}(K) \backslash \{\mathbf{0}_d\}} \left\{ \frac{\|\boldsymbol{X}^B u\|_2}{\sqrt{n}\|u_K\|_2} \right\} > 0$$

with $\mathscr{C}_{\mathrm{TV},\hat\omega}(K) = \Big\{u \in \mathbb{R}^d : \sum_{j=1}^p \|(u_{j,\bullet})_{K_j^\complement}\|_{\mathrm{TV},\hat\omega_{j,\bullet}} \leq 2 \sum_{j=1}^p \|(u_{j,\bullet})_{K_j}\|_{\mathrm{TV},\hat\omega_{j,\bullet}}\Big\}$.

Fast oracle inequality for GLM with binarsity scenario

Theorem 3 [A., Bussy, Gaïffas, Guilloux]

With a high probability, any solution $\hat{\theta}$ of the penalized problem restricted on $B_d(\rho)$ fulfills the following risk bound

$$R(m_{\hat{\theta}}) - R(m^0) \leq (1+\zeta) \inf_{\substack{\theta \in B_d(\rho) \\ \forall j \; \mathbf{1}^\top \theta_{j,\bullet} = 0 \\ |J(\theta)| \leq J^\star}} \Big\{ R(m_\theta) - R(m^0) + \frac{\xi |J(\theta)|}{\kappa^2(J(\theta))} \max_{j=1,\ldots,p} \|(\hat{\omega}_{j,\bullet})_{J_j(\theta)}\|_\infty^2 \Big\},$$

where $B_d(\rho) = \{\theta \in \mathbb{R}^d : \|\theta\|_2 \leq \rho\}$, $\zeta = Cst(C_n, \rho, p, L_n, U_n) << 1$ and $\xi = Cst(C_n, \rho, p, L_n, U_n)$.

- Since Binarsity is separable by blocks, we have

$$\big(\operatorname{prox}_{\text{bina}}(\theta)\big)_{j,\bullet} = \operatorname{prox}_{(\|\cdot\|_{\text{TV},\hat{\omega}_{j,\bullet}}+\delta_1)}(\theta_{j,\bullet}),$$

for all $j = 1, \ldots, p$.

- Algorithm 2 expresses $\operatorname{prox}_{\text{bina}}$ based on the proximal operator of the weighted TV penalization.

Algorithm 2:

Input: vector $\theta \in \mathbb{R}^d$ and weights $\hat{\omega}_{j,k}$ for $j = 1, \ldots, p$ and $k = 1, \ldots, d_j$

Output: vector $\eta = \operatorname{prox}_{\text{bina}}(\theta)$

for $j = 1$ **to** p **do**

$\beta_{j,\bullet} \leftarrow \operatorname{prox}_{\|\theta_{j,\bullet}\|_{\text{TV},\hat{\omega}_{j,\bullet}}}(\theta_{j,\bullet})$ (TV-weighted prox in block j)

$\eta_{j,\bullet} \leftarrow \beta_{j,\bullet} - \frac{1}{d_j}\sum_{k=1}^{d_j} \beta_{j,k}$ (within-block centering)

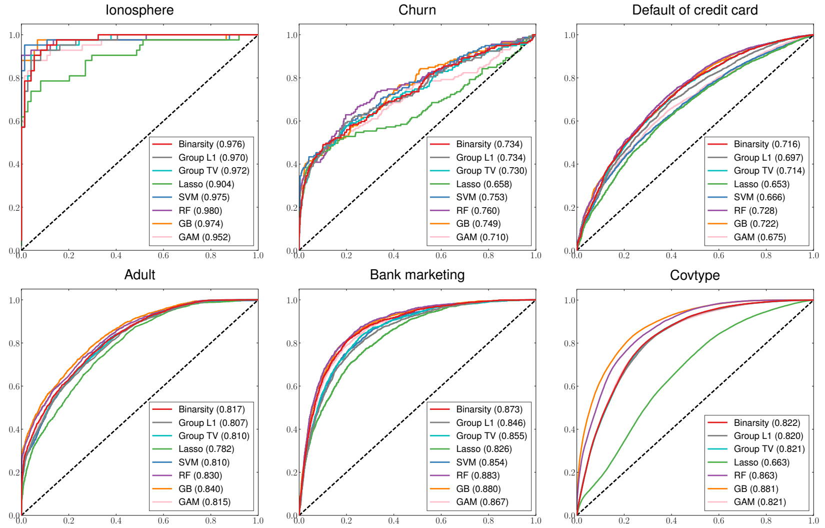
Return: η

Real data

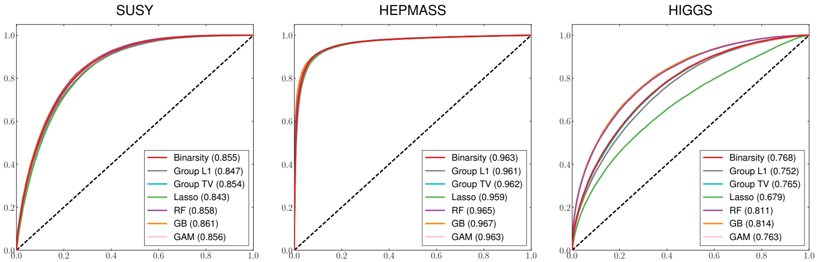
Dataset	#Samples	#Features
Ionosphere	351	34
Churn	3333	21
Default of credit card	30000	24
Adult	32561	14
Bank marketing	45211	17
Covertype	550088	10
SUSY	5000000	18
HEPMASS	10500000	28
HIGGS	11000000	24

[Source: UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/)]

Real data

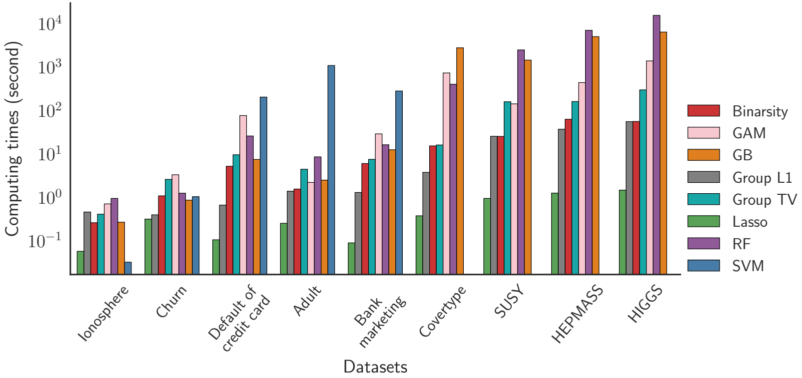


Ionosphere
Binarsity (0.976)
Group L1 (0.970)
Group TV (0.972)
Lasso (0.904)
SVM (0.975)
RF (0.980)
GB (0.974)
GAM (0.952)
Churn
Binarsity (0.734)
Group L1 (0.734)
Group TV (0.730)
Lasso (0.658)
SVM (0.753)
RF (0.760)
GB (0.749)
GAM (0.710)
Default of credit card
Binarsity (0.716)
Group L1 (0.697)
Group TV (0.714)
Lasso (0.653)
SVM (0.666)
RF (0.728)
GB (0.722)
GAM (0.675)
Adult
Binarsity (0.817)
Group L1 (0.807)
Group TV (0.810)
Lasso (0.782)
SVM (0.810)
RF (0.830)
GB (0.840)
GAM (0.815)
Bank marketing
Binarsity (0.873)
Group L1 (0.846)
Group TV (0.855)
Lasso (0.826)
SVM (0.854)
RF (0.883)
GB (0.880)
GAM (0.867)
Covtype
Binarsity (0.822)
Group L1 (0.820)
Group TV (0.821)
Lasso (0.663)
RF (0.863)
GB (0.881)
GAM (0.821)



Performance comparison using ROC and AUC scores computed on test sets. Binaristy consistnesly does a better job than Lasso, Group L1, Group TV, and GAM. Its performance is comparable to SVM, RF, and GB.

Computing time comparisons



Log-scaled computing time comparisons between the methods on the considered datasets. Binarsity is between 2 and 5 times slower than Lasso but more than 100 times faster than RF and GB on larges datasets like HIGGS.

Take Home Message

- We introduced the binarsity penalization for one-hot encodings of continuous features.
- We illustrated the good statistical properties of binarsity for generalized linear models by proving non-asymptotic oracle inequalities.
- We conducted extensive comparisons of binarsity with state-of-the-art algorithms for binary classification on several standard datasets.

Alaya, M. Z., S. Bussy, S. Gaïffas, and A. Guilloux (2017). Binarsity: a penalization for one-hot encoded features. *in revision in JMLR*.

Alaya, M. Z., S. Gaïffas, and A. Guilloux (2015). Learning the intensity of time events with change-points. *Information Theory, IEEE Transactions on 61*(9), 5148–5171.

Bacry, E., M. Bompaire, P. Deegan, S. Gaïffas, and S. V. Poulsen (2018). tick: a python library for statistical learning, with an emphasis on hawkes processes and time-dependent models. *Journal of Machine Learning Research 18*(214), 1–5.

Beck, A. and M. Teboulle (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences 2*(1), 183–202.

Boyd, S. and L. Vandenberghe (2004). *Convex optimization*. Cambridge: Cambridge University Press.

Chambolle, A., V. Caselles, D. Cremers, M. Novaga, and T. Pock (2010). An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery 9*, 263–340.

Condat, L. (2013). A Direct Algorithm for 1D Total Variation Denoising. *IEEE Signal Processing Letters 20*(11), 1054–1057.

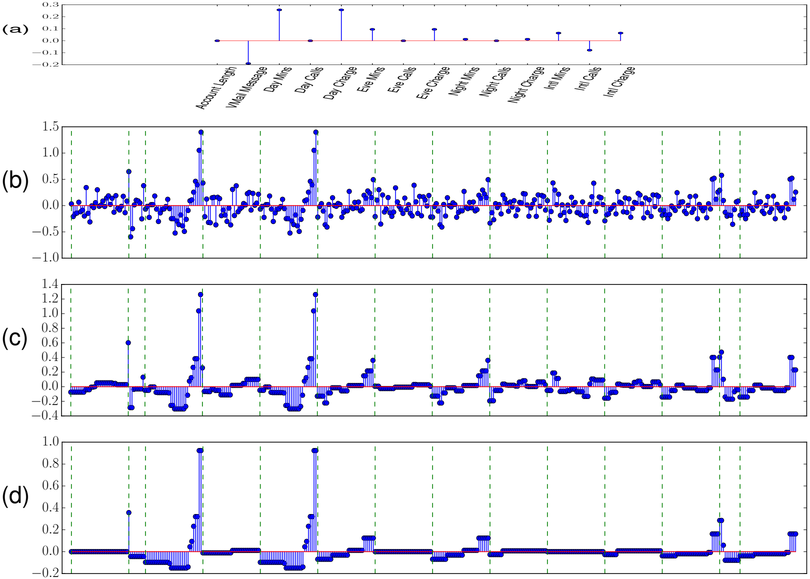
Daubechies, I., M. Defrise, and C. De Mol (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics 57*(11), 1413–1457.

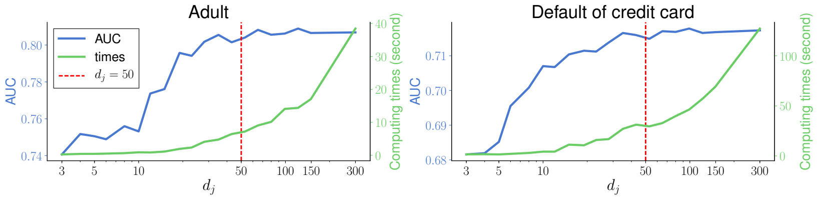
Tibshirani, R., M. Saunders, S. Rosset, J. Zhu, and K. Knight (2005). Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67*(1), 91–108.

Thank you!

Bonus

Illustration of the binarsity penalty on the "Churn" dataset.





Impact of the number of bins used in each block (d_j) on the classification performance (measured by AUC) and on the training time using the "Adult" and "Default of credit card" datasets. We observe that past $d_j = 50$ bins, performance is roughly constant, while training time strongly increases.