Large-scale semi-supervised learning with online spectral graph sparsification

daniele.calandriello@inria.fr, alessandro.lazaric@inria.fr, michal.valko@inria.fr

Graph Learning

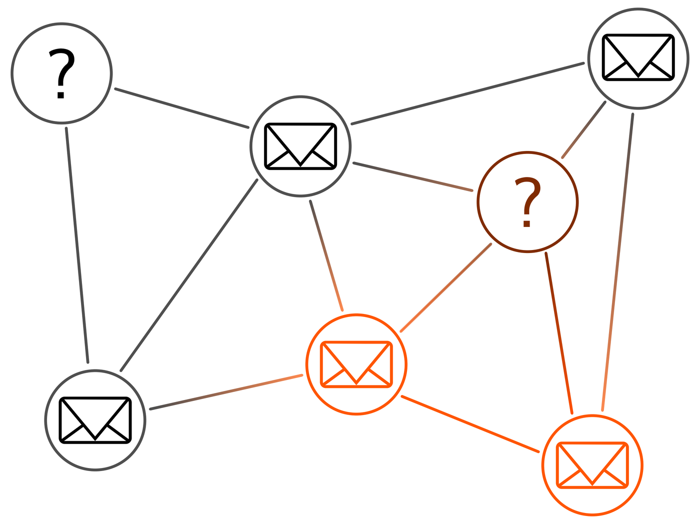
Draw $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ from $\mathbb{R}^d$,
Build the graph $\mathcal{G} = (\mathcal{X}, \mathcal{E})$ with $|\mathcal{E}| = m$,
The weights $a_{e_{i,j}}$ encode the "distance" between nodes.

Transductive setting for Semi-Supervised Learning

There exists a label y_i for each node in $\mathcal{G}$
l nodes are placed in $\mathcal{S}$ the remaining $u = n - l$ in $\mathcal{T}$
The algorithm receive the labels in $\mathcal{S}$ and the graph $\mathcal{G}$ and outputs a function $\mathbf{f} : \mathcal{X} \to \mathbb{R}$.
Measure $\mathbf{f}$ prediction error over $\mathcal{T}$
The graph $\mathcal{G}$ never changes
↳ (e.g. in spam classification, our email corpus is fixed)
The subset $\mathcal{S}$ that is revealed to the algorithm is random
↳ (e.g. which emails the users classify as spam or ham)

Harmonic Function Solution (HFS)

$$\widehat{\mathbf{f}} = \arg\min_{\mathbf{f} \in \mathbb{R}^n} \frac{1}{l}(\mathbf{f} - \mathbf{y})^\mathsf{T} I_{\mathcal{S}} (\mathbf{f} - \mathbf{y}) + \gamma \mathbf{f}^\mathsf{T} L_{\mathcal{G}} \mathbf{f}, \qquad (1)$$



Algorithmic Stability

$$\begin{matrix} \mathcal{S} \\ \mathcal{S}' \end{matrix} = \begin{pmatrix} y_1, y_2, y_3, y_4, \ldots, y_{l-1}, y_l, & 0, & 0, 0, 0, \ldots \\ y_1, y_2, y_3, y_4, \ldots, y_{l-1}, & 0, & y_{l+1}, 0, 0, 0, \ldots \end{pmatrix} \to \begin{matrix} \mathbf{f} \\ \mathbf{f}' \end{matrix}$$

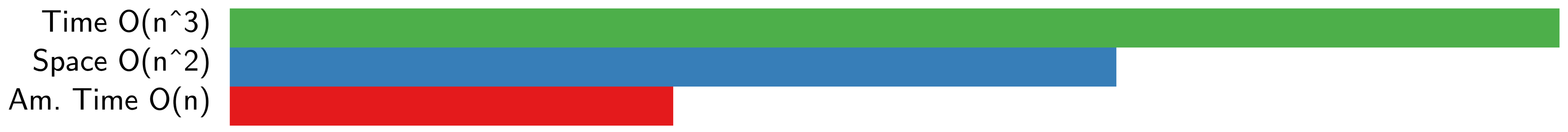
$$|(\mathbf{f}(\mathbf{x}) - \mathbf{y}(\mathbf{x}))^2 - (\mathbf{f}'(\mathbf{x}) - \mathbf{y}(\mathbf{x}))^2| \le \beta.$$

Theoretical guarantees for stable transductive algorithms [1]

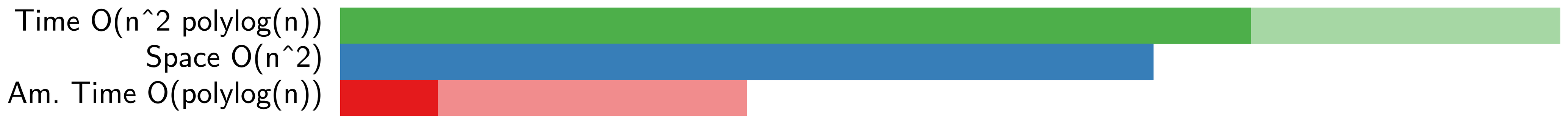
$$R(\widetilde{\mathbf{f}}) \le \widehat{R}(\widetilde{\mathbf{f}}) + \beta + \left(2\beta + \frac{c^2(l+u)}{lu}\right)\sqrt{\frac{\pi(l,u)\log(1/\delta)}{2}}$$

$$\pi(l,u) = \frac{lu}{l+u-0.52\max\{l,u\}} \frac{2\max\{l,u\}}{-1}$$

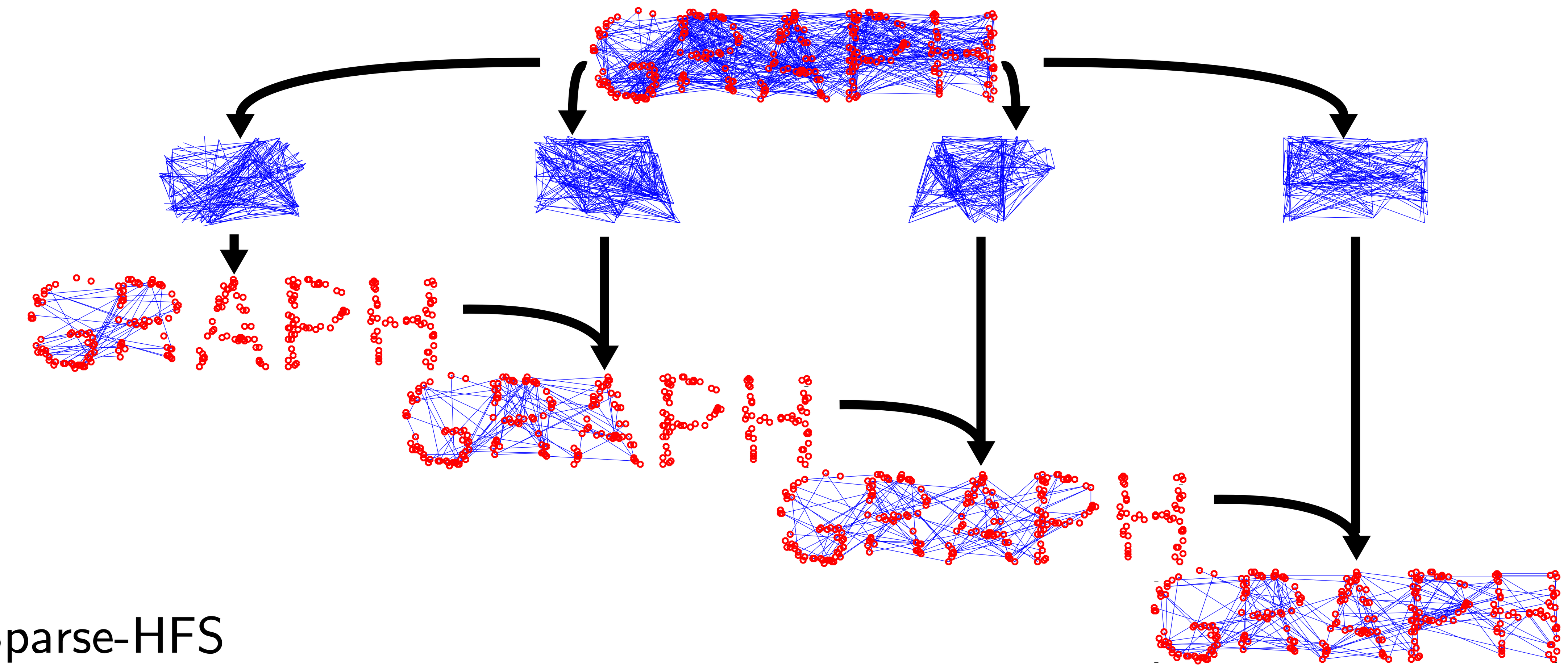
Stable-HFS [2]: $\mu = ((\gamma l L_{\mathcal{G}} + I_{\mathcal{S}})^{+} \mathbf{y}_{\mathcal{S}})^\mathsf{T} \mathbf{1} / ((\gamma l L_{\mathcal{G}} + I_{\mathcal{S}})^{+} \mathbf{1})^\mathsf{T} \mathbf{1}$, $\widehat{\mathbf{f}} = (\gamma l L_{\mathcal{G}} + I_{\mathcal{S}})^{+}(\widetilde{\mathbf{y}}_{\mathcal{S}} - \mu \mathbf{1}) = \left(P_{\mathcal{F}}(\gamma l L_{\mathcal{G}} + I_{\mathcal{S}})\right)^{+} \widetilde{\mathbf{y}}_{\mathcal{S}}$



Fast SDD Linear Solver [3] + Stable-HFS



Laplacian spectral ε-sparsifier: $(1-\varepsilon)\mathbf{x}^\mathsf{T} L_{\mathcal{G}} \mathbf{x} \le \mathbf{x}^\mathsf{T} L_{\mathcal{H}} \mathbf{x} \le (1+\varepsilon)\mathbf{x}^\mathsf{T} L_{\mathcal{G}} \mathbf{x}.$



Sparse-HFS

```
input  Graph 𝒢 = (𝒳, ℰ), labels y_𝒮, accuracy ε
output  Solution f̂, sparsified graph ℋ
  Let α = 1/(1 − ε) and N = α² n log²(n)/ε²
  Partition ℰ in τ = ⌈m/N⌉ blocks Δ_1, ..., Δ_τ
  Initialize ℋ = ∅
  for t = 1, ..., τ do
    Load Δ_t in memory
    Compute ℋ_t = SPARSIFY(ℋ_{t−1}, Δ_t, N, α)
  end for
  Center the labels ỹ_𝒮
  Compute f̃ with STABLE-HFS with ỹ_𝒮 using a
  suitable SDD solver
```

Figure 1: SPARSE-HFS

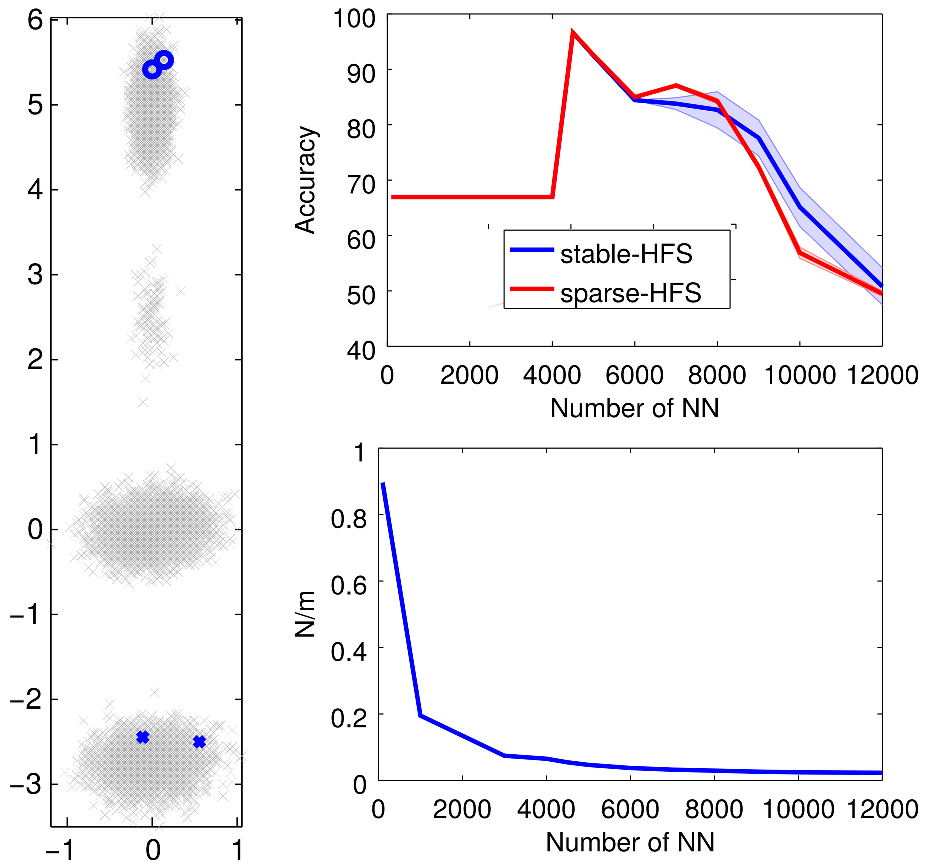
```
input  A sparsifier ℋ, block Δ, number of edges N, effective resistance accuracy α
output  A sparsifier ℋ′, probabilities {p̃′_e : e ∈ ℋ′}.
  Compute estimates of R̃′_e for any edge in ℋ + Δ such that 1/α ≤ R̃′_e/R′_e ≤ α with an SDD solver [13]
  Compute probabilities p̃′_e = (a_e R̃′_e)/(α(n − 1)) and weights w_e = a_e/(N p̃′_e)
  For all edges e ∈ ℋ compute p̃′_e ← min{p̃_e, p̃′_e} and initialize ℋ′ = ∅
  for all edges e ∈ ℋ do
    Add edge e to ℋ′ with weight w_e with probability p̃′_e/p̃_e
  end for
  for all edges e ∈ Δ do
    for i = 1 to N do
      Add edge e to ℋ′ with weight w_e with probability p̃′_e
    end for
  end for
```

Figure 2: Kelner-Levin Sparsification Algorithm [12]

$$R(\widetilde{\mathbf{f}}) \le \widehat{R}(\widetilde{\mathbf{f}}) + \frac{l^2\gamma^2\lambda_n(\mathcal{G})^2 M^2 \varepsilon^2}{(l\gamma(1-\varepsilon)\lambda_2(\mathcal{G})-1)^4} + \beta + \left(2\beta + \frac{c^2(l+u)}{lu}\right)\sqrt{\frac{\pi(l,u)\ln\frac{1}{\delta}}{2}},$$

$$\beta \le \frac{1.5M\sqrt{l}}{(l\gamma(1-\varepsilon)\lambda_2(\mathcal{G})-1)^2} + \frac{4M}{l\gamma(1-\varepsilon)\lambda_2(\mathcal{G})-1}$$

Toy example

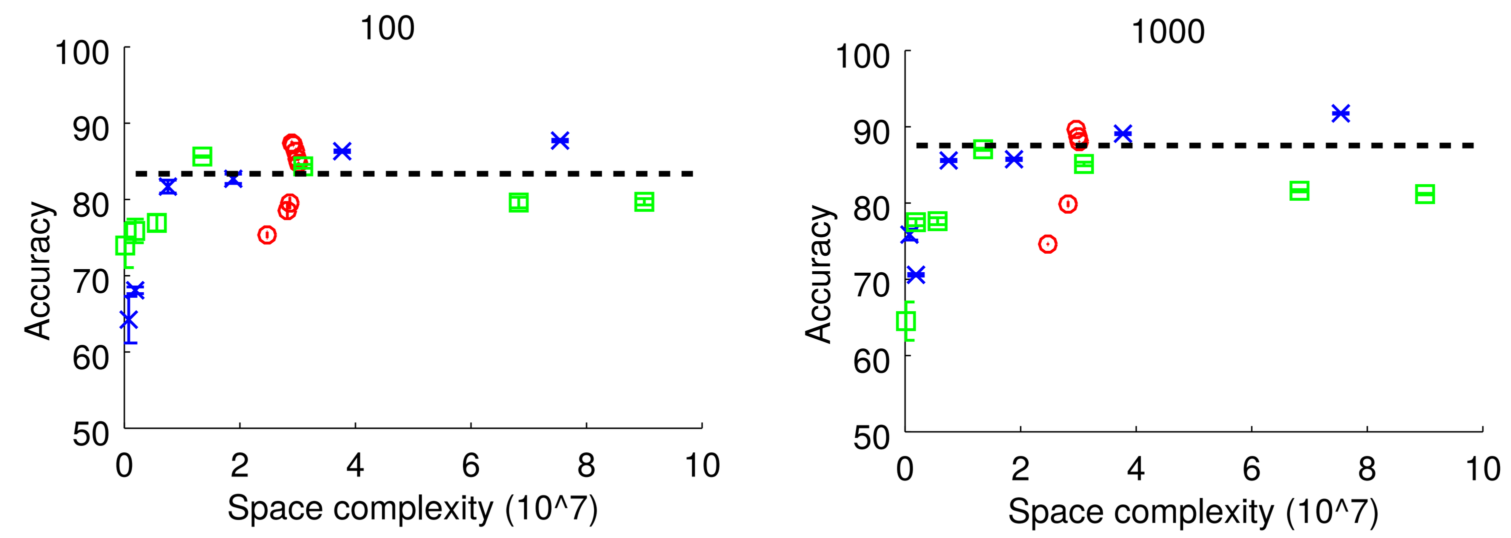


Spam Classification (TREC07)

	Guarantees	Space	Preprocessing Time	Solving Time
SparseHFS	✔	$\mathcal{O}(n\,\text{polylog}(n))$	$\mathcal{O}(m\,\text{polylog}(n))$	$\mathcal{O}(n\,\text{polylog}(n))$
StableHFS	✔	$\mathcal{O}(n^2)$	$\mathcal{O}(m)$	$\mathcal{O}(m\,\text{polylog}(n))$
Fergus	✘	$\mathcal{O}(nd + nk + b^2)$	$\mathcal{O}(kb^3 + db^3)$	$\mathcal{O}(k^2\,\text{polylog}(k) + nk)$
SimpleHFS	✔	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(k^3)$
SubSample	✘	$\mathcal{O}(k^2)$	$\mathcal{O}(m)$	$\mathcal{O}(k^2\,\text{polylog}(k) + n)$

✔ (under assumptions)

$n = 75419$ raw emails
$y =$ HAM or SPAM
$d = 68697$ features, TF-IDF extracted from the text
$l = \{100, 1000\}$
$\varepsilon = 0.8$



ExtraLearn

We would like to thank Ioannis Koutis for many useful discussions.

[1] C. Cortes, M. Mohri, D. Pechyony, and A. Rastogi. Stability of transductive regression algorithms. ICML 2008.
[2] M. Belkin, I. Matveeva, and P. Niyogi. Regularization and Semi-Supervised Learning on Large Graphs. COLT, 2004
[3] I. Koutis, G. L. Miller, and R. Peng. A nearly-m log n time solver for SDD linear systems. FOCS, 2011