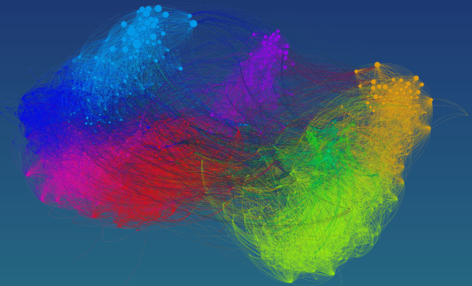
Graphs in Machine Learning

Michal Valko

Inria Lille - Nord Europe, France

TA: Pierre Perrault

Partially based on material by: Mikhail Belkin, Jerry Zhu, Olivier Chapelle, Branislav Kveton



Previous Lecture

- recommendation on a bipartite graph
- resistive networks
  - recommendation score as a resistance?
  - Laplacian and resistive networks
  - resistance distance and random walks
- geometry of the data and the connectivity
- spectral clustering
  - connectivity vs. compactness
  - MinCut, RatioCut, NCut
  - spectral relaxations
- manifold learning with Laplacian eigenmaps

Previous Lab Session

- 23. 10. 2017 by Pierre Perrault
- Content
  - graph construction
  - test sensitivity to parameters: σ, k, ε
  - spectral clustering
  - spectral clustering vs. k-means
  - image segmentation
- Short written report (graded, all reports around 40% of grade)
- Check the course website for the policies
- Questions to piazza
- *Deadline:* 6. 11. 2017, 23:59

This Lecture

- manifold learning with Laplacian eigenmaps
- semi-supervised learning
- inductive and transductive semi-supervised learning
- SSL with self-training
- SVMs and semi-supervised SVMs = TSVMs
- Gaussian random fields and harmonic solution
- graph-based semi-supervised learning
- transductive learning
- manifold regularization

Manifold Learning: Recap

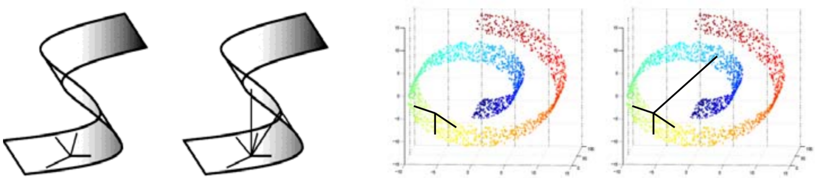
problem: definition reduction/manifold learning

Given $\{\mathbf{x}_i\}_{i=1}^{N}$ from $\mathbb{R}^d$ find $\{\mathbf{y}_i\}_{i=1}^{N}$ in $\mathbb{R}^m$, where $m \ll d$.

- What do we know about the **dimensionality reduction**
  - representation/visualization (2D or 3D)
  - an old example: globe to a map
  - often assuming $\mathcal{M} \subset \mathbb{R}^d$
  - feature extraction
  - linear vs. nonlinear dimensionality reduction
- What do we know about linear vs. nonlinear methods?
  - linear: ICA, PCA, SVD, ...
  - nonlinear often preserve only **local** distances

Manifold Learning: Linear vs. Non-linear

Manifold Learning: Preserving (just) local distances



$$d(\mathbf{y}_i, \mathbf{y}_j) = d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{only if} \quad d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{is small}$$

$$\min \sum_{ij} w_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2$$

Looks familiar?

Manifold Learning: Laplacian Eigenmaps

Step 1: Solve generalized eigenproblem:

$$\mathbf{Lf} = \lambda \mathbf{Df}$$

Step 2: Assign m new coordinates:

$$\mathbf{x}_i \mapsto \left(f_2\left(i\right), \ldots, f_{m+1}\left(i\right)\right)$$

Note$_1$: we need to get $m+1$ smallest eigenvectors
Note$_2$: $\mathbf{f}_1$ is useless

http://web.cse.ohio-state.edu/~mbelkin/papers/LEM_NC_03.pdf

Manifold Learning: Laplacian Eigenmaps to 1D

Laplacian Eigenmaps 1D objective

$$\min_{\mathbf{f}} \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} \quad \text{s.t.} \quad f_i \in \mathbb{R}, \quad \mathbf{f}^\mathsf{T}\mathbf{D}\mathbf{1} = 0, \quad \mathbf{f}^\mathsf{T}\mathbf{D}\mathbf{f} = \mathbf{1}$$

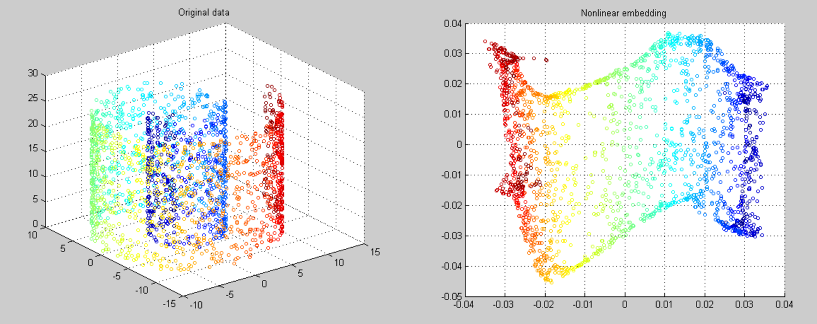
The meaning of the constraints is similar as for spectral clustering:

$\mathbf{f}^\mathsf{T}\mathbf{D}\mathbf{f} = \mathbf{1}$ is for scaling

$\mathbf{f}^\mathsf{T}\mathbf{D}\mathbf{1} = 0$ is to not get $\mathbf{v}_1$

What is the solution?

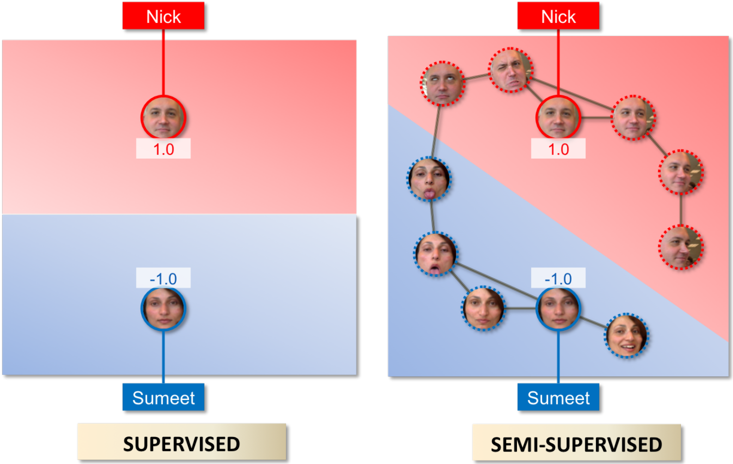
Manifold Learning: Example



http://www.mathworks.com/matlabcentral/fileexchange/

36141-laplacian-eigenmap-~-diffusion-map-~-manifold-learning

Semi-supervised learning: How is it possible?



This is how children learn! hypothesis

Semi-supervised learning (SSL)

SSL problem: definition

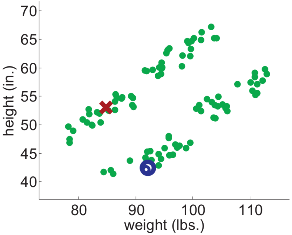
Given $\{\mathbf{x}_i\}_{i=1}^{N}$ from $\mathbb{R}^d$ and $\{y_i\}_{i=1}^{n_l}$, with $n_l \ll N$, find $\{y_i\}_{i=n_l+1}^{n}$ (**transductive**) or find f predicting y well beyond that (**inductive**).

Some facts about SSL

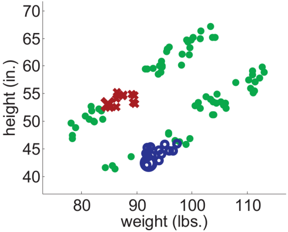
- assumes that the unlabeled data is useful
- works with data geometry assumptions
  - cluster assumption — low-density separation
  - manifold assumption
  - smoothness assumptions, generative models, ...
- now it helps now, now it does not (sic)
  - provable cases when it helps
- inductive or transductive/out-of-sample extension

http://olivier.chapelle.cc/ssl-book/discussion.pdf

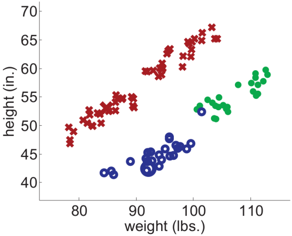
SSL: Self-Training



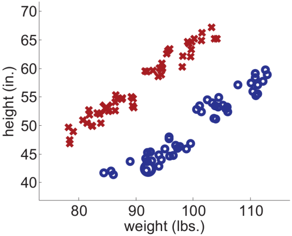
(a) Iteration 1



(b) Iteration 25



(c) Iteration 74



(d) Final labeling of all instances

SSL: Overview: Self-Training

SSL: Self-Training

Input: $\mathcal{L} = \{\mathbf{x}_i, y_i\}_{i=1}^{n_l}$ and $\mathcal{U} = \{\mathbf{x}_i\}_{i=n_l+1}^{N}$

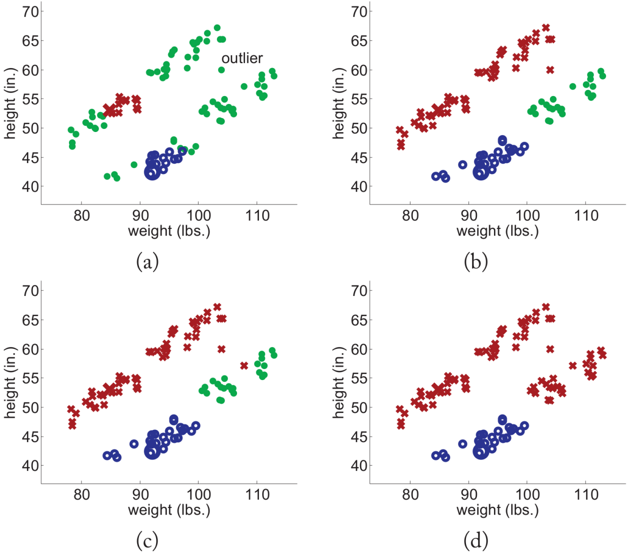
Repeat:

- train f using $\mathcal{L}$
- apply f to (some) $\mathcal{U}$ and add them to $\mathcal{L}$

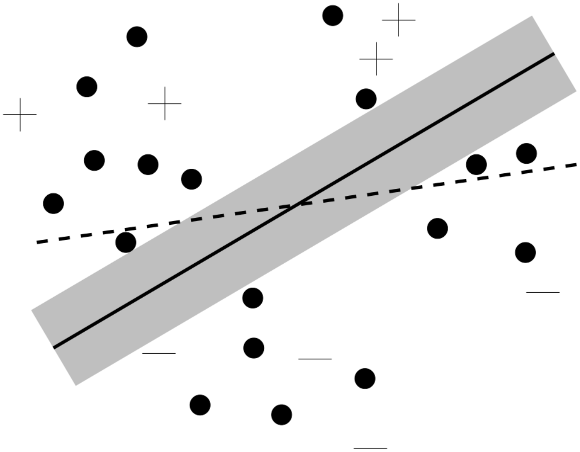
What are the properties of self-training?

- its a wrapper method
- heavily depends on the the internal classifier
- some theory exist for specific classifiers
- nobody uses it anymore
- errors propagate (unless the clusters are well separated)

SSL: Self-Training: Bad Case



SSL: Transductive SVM: S3VM



SSL: Transductive SVM: Classical SVM

Linear case: $f = \mathbf{w}^\mathsf{T}\mathbf{x} + b \quad \rightarrow \quad$ we look for $(\mathbf{w}, b)$

max-margin classification

$$\max_{\mathbf{w},b} \quad \frac{1}{\|\mathbf{w}\|}$$
$$s.t. \quad y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 \quad \forall i = 1, \ldots, n_l$$

note the difference between functional and geometric margin

max-margin classification

$$\min_{\mathbf{w},b} \quad \|\mathbf{w}\|^2$$
$$s.t. \quad y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 \quad \forall i = 1, \ldots, n_l$$

SSL: Transductive SVM: Classical SVM

max-margin classification: separable case

$$\min_{\mathbf{w},b} \quad \|\mathbf{w}\|^2$$
$$s.t. \quad y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 \quad \forall i = 1, \ldots, n_l$$

max-margin classification: non-separable case

$$\min_{\mathbf{w},b} \quad \lambda\|\mathbf{w}\|^2 + \sum_i \xi_i$$
$$s.t. \quad y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, \ldots, n_l$$
$$\xi_i \geq 0 \quad \forall i = 1, \ldots, n_l$$

SSL: Transductive SVM: Classical SVM

max-margin classification: non-separable case

$$\min_{\mathbf{w},b} \quad \lambda\|\mathbf{w}\|^2 + \sum_i \xi_i$$

$$s.t. \quad y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1, \dots, n_l$$

$$\xi_i \geq 0 \quad \forall i = 1, \dots, n_l$$

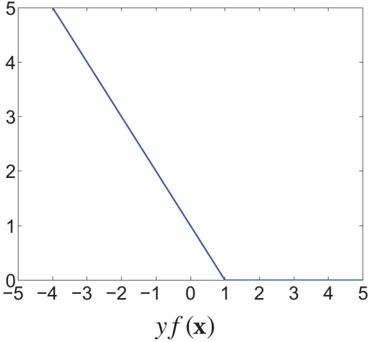
Unconstrained formulation using **hinge loss**:

$$\min_{\mathbf{w},b} \sum_i^{n_l} \max\left(1 - y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right), 0\right) + \lambda\|\mathbf{w}\|^2$$

In general?

$$\min_{\mathbf{w},b} \sum_i^{n_l} V(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \lambda\Omega(f)$$

SSL: Transductive SVM: Classical SVM: Hinge loss



(a) the hinge loss

$$V(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \max\left(1 - y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right), 0\right)$$

SSL: Transductive SVM: Unlabeled Examples

$$\min_{\mathbf{w},b} \sum_i^{n_l} \max\left(1 - y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right), 0\right) + \lambda\|\mathbf{w}\|^2$$

How to incorporate unlabeled examples?

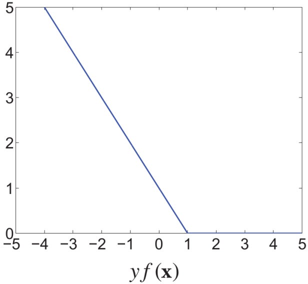
No y's for unlabeled $\mathbf{x}$.

Prediction of f for (any) $\mathbf{x}$? $\widehat{y} = \operatorname{sgn}\left(f\left(\mathbf{x}\right)\right) = \operatorname{sgn}\left(\mathbf{w}^\mathsf{T}\mathbf{x} + b\right)$

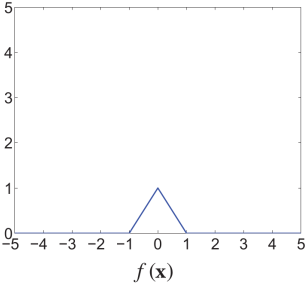
Pretending that $\operatorname{sgn}\left(f\left(\mathbf{x}\right)\right)$ is the true label ...

$$\begin{aligned}
V(\mathbf{x}, \widehat{y}, f(\mathbf{x})) &= \max\left(1 - \widehat{y}\left(\mathbf{w}^\mathsf{T}\mathbf{x} + b\right), 0\right) \\
&= \max\left(1 - \operatorname{sgn}\left(\mathbf{w}^\mathsf{T}\mathbf{x} + b\right)\left(\mathbf{w}^\mathsf{T}\mathbf{x} + b\right), 0\right) \\
&= \max\left(1 - \left|\mathbf{w}^\mathsf{T}\mathbf{x} + b\right|, 0\right)
\end{aligned}$$

SSL: Transductive SVM: Hinge and Hat Loss



(a) the hinge loss



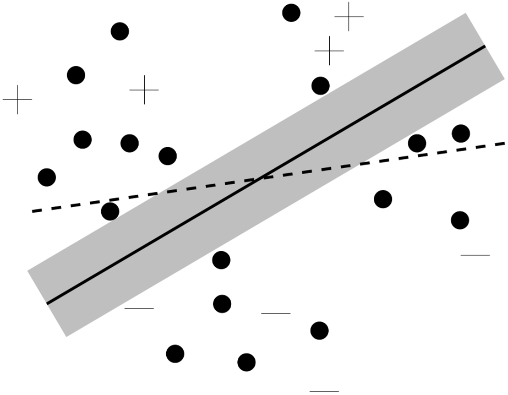
(b) the hat loss

What is the difference in the objectives?

Hinge loss penalizes?

Hat loss penalizes?

SSL: Transductive SVM: S3VM



This is what we wanted!

SSL: Transductive SVM: Formulation

Main SVM idea stays the same: penalize the margin

$$\min_{\mathbf{w},b} \sum_{i=1}^{n_l} \max\left(1 - y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right), 0\right) + \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{i=n_l+1}^{n_l+n_u} \max\left(1 - \left|\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right|, 0\right)$$

What is the loss and what is the regularizer?

$$\min_{\mathbf{w},b} \sum_{i=1}^{n_l} \max\left(1 - y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right), 0\right) + \lambda_1 \|\mathbf{w}\|^2 + \lambda_2 \sum_{i=n_l+1}^{n_l+n_u} \max\left(1 - \left|\mathbf{w}^\mathsf{T}\mathbf{x}_i + b\right|, 0\right)$$

Think of **unlabeled data** as the **regularizers** for your classifiers!

Practical hint: Additionally enforce the class balance.

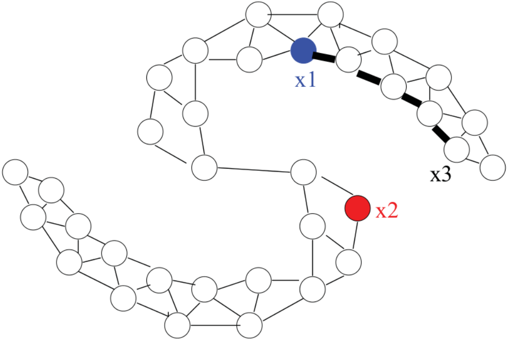
What it the main issue of TSVM?

recent advancements: `http://jmlr.org/proceedings/papers/v48/hazanb16.pdf`

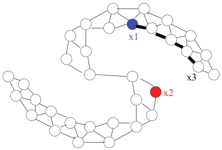
SSL with Graphs: Prehistory

Blum/Chawla: Learning from Labeled and Unlabeled Data using Graph Mincuts
`http://www.aladdin.cs.cmu.edu/papers/pdfs/y2001/mincut.pdf`

*following some insights from vision research in 1980s



SSL with Graphs: MinCut



MinCut SSL: an idea similar to MinCut clustering

Where is the link?

What is the formal statement? We look for $f(\mathbf{x}) \in \{\pm 1\}$

$$\mathrm{cut} = \sum_{i,j=1}^{n_l+n_u} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2 = \Omega(f)$$

Why $\left(f\left(\mathbf{x}_i\right) - f\left(\mathbf{x}_j\right)\right)^2$ and not $|f(\mathbf{x}_i) - f(\mathbf{x}_j)|$?

SSL with Graphs: MinCut

We look for $f(\mathbf{x}) \in \{\pm 1\}$

$$\Omega(\mathbf{f}) = \sum_{i,j=1}^{n_l+n_u} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2$$

Clustering was unsupervised, here we have supervised data.

Recall the general objective-function framework:

$$\min_{\mathbf{w},b} \sum_{i}^{n_l} V(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \lambda \Omega(\mathbf{f})$$

It would be nice if we match the prediction on labeled data:

$$V(\mathbf{x}, y, f(\mathbf{x})) = \infty \sum_{i=1}^{n_l} \left(f(\mathbf{x}_i) - y_i\right)^2$$

SSL with Graphs: MinCut

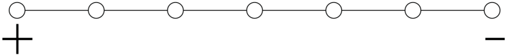
Final objective function:

$$\min_{\mathbf{f}\in\{\pm 1\}^{n_l+n_u}} \infty \sum_{i=1}^{n_l} (f(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{i,j=1}^{n_l+n_u} w_{ij} (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2$$

This is an integer program :(

Can we solve it?

Are we happy?



We need a better way to reflect the confidence.

SSL with Graphs: Harmonic Functions

Zhu/Ghahramani/Lafferty: Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions

http://mlg.eng.cam.ac.uk/zoubin/papers/zgl.pdf

*a seminal paper that convinced people to use graphs for SSL

Idea 1: Look for a **unique** solution.

Idea 2: Find a smooth one. (**harmonic** solution)

Harmonic SSL

1): As before we constrain f to match the supervised data:

$$f(\mathbf{x}_i) = y_i \qquad \forall i \in \{1, \ldots, n_l\}$$

2): We enforce the solution f to be harmonic.

$$f(\mathbf{x}_i) = \frac{\sum_{i \sim j} f(\mathbf{x}_j) w_{ij}}{\sum_{i \sim j} w_{ij}} \qquad \forall i \in \{n_l + 1, \ldots, n_u + n_l\}$$

SSL with Graphs: Harmonic Functions

The harmonic solution is obtained from the mincut one ...

$$\min_{\mathbf{f}\in\{\pm 1\}^{n_l+n_u}} \infty \sum_{i=1}^{n_l} (f(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{i,j=1}^{n_l+n_u} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2$$

... if we just relax the integer constraints to be real ...

$$\min_{\mathbf{f}\in\mathbb{R}^{n_l+n_u}} \infty \sum_{i=1}^{n_l} (f(\mathbf{x}_i) - y_i)^2 + \lambda \sum_{i,j=1}^{n_l+n_u} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2$$

... or equivalently (note that $f(\mathbf{x}_i) = f_i$) ...

$$\min_{\mathbf{f}\in\mathbb{R}^{n_l+n_u}} \sum_{i,j=1}^{n_l+n_u} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2$$

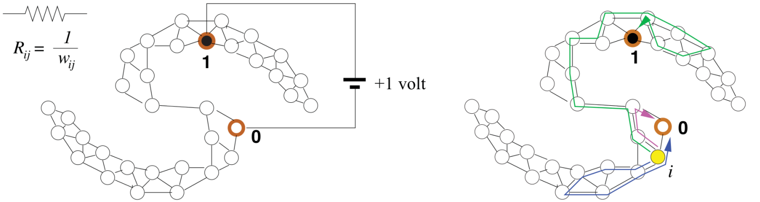
$$s.t. \quad y_i = f(\mathbf{x}_i) \quad \forall i = 1, \dots, n_l$$

SSL with Graphs: Harmonic Functions

Properties of the relaxation from ± 1 to $\mathbb{R}$

- there is a closed form solution for $\mathbf{f}$
- this solution is unique
- globally optimal
- it is either constant or has a maximum/minimum on a boundary
- $f(\mathbf{x}_i)$ may not be discrete
  - but we can threshold it
- electric-network interpretation
- random-walk interpretation

SSL with Graphs: Harmonic Functions



(a) The electric network interpretation (b) The random walk interpretation

Random walk interpretation:

1) start from the vertex you want to label and randomly walk

2) $P(j|i) = \frac{w_{ij}}{\sum_k w_{ik}} \qquad \equiv \qquad \mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$

3) finish when **a** labeled vertex is hit

absorbing random walk

$f_i =$ probability of reaching a positive labeled vertex

SSL with Graphs: Harmonic Functions

How to compute HS? **Option A:** iteration/propagation

Step 1: Set $f(\mathbf{x}_i) = y_i$ for $i = 1, \ldots, n_l$
Step 2: Propagate iteratively (only for unlabeled)

$$f(\mathbf{x}_i) \leftarrow \frac{\sum_{i\sim j} f(\mathbf{x}_j) w_{ij}}{\sum_{i\sim j} w_{ij}} \qquad \forall i \in \{n_l + 1, \ldots, n_u + n_l\}$$

Properties:

- this will converge to the harmonic solution
- we can set the initial values for unlabeled nodes arbitrarily
- an interesting option for large-scale data

SSL with Graphs: Harmonic Functions

How to compute HS? **Option B:** Closed form solution

Define $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_{n_l+n_u})) = (f_1, \ldots, f_{n_l+n_u})$

$$\Omega(\mathbf{f}) = \sum_{i,j=1}^{n_l+n_u} w_{ij} \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2 = \mathbf{f}^{\mathsf{T}}\mathbf{L}\mathbf{f}$$

$\mathbf{L}$ is a $(n_l + n_u) \times (n_l + n_u)$ matrix:

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{ll} & \mathbf{L}_{lu} \\ \mathbf{L}_{u1} & \mathbf{L}_{uu} \end{bmatrix}$$

How to compute this **constrained** minimization problem?

SSL with Graphs: Harmonic Functions

Let us compute **harmonic** solution using **harmonic** property!

How did we formalize the harmonic property of a circuit?

$$(\mathbf{Lf})_u = \mathbf{0}_u$$

In matrix notation

$$\begin{bmatrix} \mathbf{L}_{ll} & \mathbf{L}_{lu} \\ \mathbf{L}_{ul} & \mathbf{L}_{uu} \end{bmatrix} \begin{bmatrix} \mathbf{f}_l \\ \mathbf{f}_u \end{bmatrix} = \begin{bmatrix} \dots \\ \mathbf{0}_u \end{bmatrix}$$

$\mathbf{f}_l$ is constrained to be $\mathbf{y}_l$ and for $\mathbf{f}_u$ . . . . . .

$$\mathbf{L}_{ul}\mathbf{f}_l + \mathbf{L}_{uu}\mathbf{f}_u = \mathbf{0}_u$$

. . . from which we get

$$\mathbf{f}_u = \mathbf{L}_{uu}^{-1}(-\mathbf{L}_{ul}\mathbf{f}_l) = \mathbf{L}_{uu}^{-1}(\mathbf{W}_{ul}\mathbf{f}_l).$$

Note that this does not depend on $\mathbf{L}_{ll}$.

SSL with Graphs: Harmonic Functions

Can we see that this calculates the probability of a random walk?

$$\mathbf{f}_u = \mathbf{L}_{uu}^{-1}(-\mathbf{L}_{ul}\mathbf{f}_l) = \mathbf{L}_{uu}^{-1}(\mathbf{W}_{ul}\mathbf{f}_l)$$

Note that $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$. Then equivalently

$$\mathbf{f}_u = (\mathbf{I} - \mathbf{P}_{uu})^{-1}\mathbf{P}_{ul}\mathbf{f}_l.$$

Split the equation into +ve & -ve part:

$$\begin{aligned} f_i &= (\mathbf{I} - \mathbf{P}_{uu})_{iu}^{-1}\mathbf{P}_{ul}\mathbf{f}_l \\ &= \underbrace{\sum_{j:y_j=1} (\mathbf{I} - \mathbf{P}_{uu})_{iu}^{-1}\mathbf{P}_{uj}}_{p_i^{(+1)}} - \underbrace{\sum_{j:y_j=-1} (\mathbf{I} - \mathbf{P}_{uu})_{iu}^{-1}\mathbf{P}_{uj}}_{p_i^{(-1)}} \\ &= p_i^{(+1)} - p_i^{(-1)} \end{aligned}$$

SSL with Graphs: Regularized Harmonic Functions

$$f_i = p_i^{(+1)} - p_i^{(-1)} \qquad \Longrightarrow \quad f_i = \underbrace{|f_i|}_{\text{confidence}} \times \underbrace{\operatorname{sgn}(f_i)}_{\text{label}}$$

What if a nasty outlier sneaks in?

The prediction for the outlier can be hyperconfident :(

How to control the confidence of the inference?

Allow the random walk to **die**!

We add a **sink** to the graph.

sink = artificial label node with value 0

We connect it to every other vertex.

What will this do to our predictions?

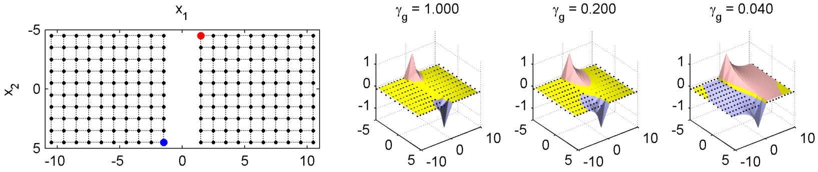
depends on the weigh on the edges

SSL with Graphs: Regularized Harmonic Functions

How do we compute this **regularized** random walk?

$$\mathbf{f}_u = (\mathbf{L}_{uu} + \gamma_g \mathbf{I})^{-1} (\mathbf{W}_{ul} \mathbf{f}_l)$$

How does γ_g influence HS?



What happens to sneaky outliers?

SSL with Graphs: Harmonic Functions

Why don't we represent the sink in **L** explicitly?

Formally, to get the harmonic solution on the graph with sink ...

$$\begin{bmatrix} \mathbf{L}_{ll} + \gamma_G \mathbf{I}_{n_l} & \mathbf{L}_{lu} & -\gamma_G \\ \mathbf{L}_{ul} & \mathbf{L}_{uu} + \gamma_G \mathbf{I}_{n_u} & -\gamma_G \\ -\gamma_G \mathbf{1}_{n_l \times 1} & -\gamma_G \mathbf{1}_{n_u \times 1} & n\gamma_G \end{bmatrix} \begin{bmatrix} \mathbf{f}_l \\ \mathbf{f}_u \\ 0 \end{bmatrix} = \begin{bmatrix} \dots \\ \mathbf{0}_u \\ \dots \end{bmatrix}$$

$$\mathbf{L}_{ul}\mathbf{f}_l + \left(\mathbf{L}_{uu} + \gamma_G \mathbf{I}_{n_u}\right)\mathbf{f}_u = \mathbf{0}_u$$

... which is the same if we disregard the last column and row ...

$$\begin{bmatrix} \mathbf{L}_{ll} + \gamma_G \mathbf{I}_{n_l} & \mathbf{L}_{lu} \\ \mathbf{L}_{ul} & \mathbf{L}_{uu} + \gamma_G \mathbf{I}_{n_u} \end{bmatrix} \begin{bmatrix} \mathbf{f}_l \\ \mathbf{f}_u \end{bmatrix} = \begin{bmatrix} \dots \\ \mathbf{0}_u \end{bmatrix}$$

... and therefore we simply add γ_G to the diagonal of **L**!

SSL with Graphs: Soft Harmonic Functions

Regularized HS objective with $\mathbf{Q} = \mathbf{L} + \gamma_g \mathbf{I}$:

$$\min_{\mathbf{f} \in \mathbb{R}^{n_l + n_u}} \infty \sum_{i=1}^{n_l} (f(\mathbf{x}_i) - y_i)^2 + \lambda \mathbf{f}^\mathsf{T} \mathbf{Q} \mathbf{f}$$

What if we do not really believe that $f(\mathbf{x}_i) = y_i$, $\forall i$?

$$\mathbf{f}^\star = \min_{\mathbf{f} \in \mathbb{R}^N} (\mathbf{f} - \mathbf{y})^\mathsf{T} \mathbf{C} (\mathbf{f} - \mathbf{y}) + \mathbf{f}^\mathsf{T} \mathbf{Q} \mathbf{f}$$

$\mathbf{C}$ is diagonal with $C_{ii} = \begin{cases} c_l & \text{for labeled examples} \\ c_u & \text{otherwise.} \end{cases}$

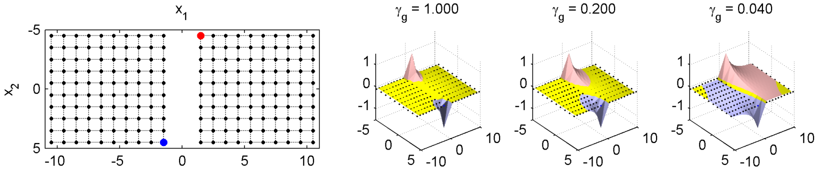
$\mathbf{y} \equiv$ pseudo-targets with $y_i = \begin{cases} \text{true label} & \text{for labeled examples} \\ 0 & \text{otherwise.} \end{cases}$

SSL with Graphs: Soft Harmonic Functions

$$\mathbf{f}^\star = \min_{\mathbf{f}\in\mathbb{R}^n} (\mathbf{f}-\mathbf{y})^\mathsf{T}\mathbf{C}(\mathbf{f}-\mathbf{y}) + \mathbf{f}^\mathsf{T}\mathbf{Q}\mathbf{f}$$

Closed form **soft harmonic solution**:

$$\mathbf{f}^\star = (\mathbf{C}^{-1}\mathbf{Q}+\mathbf{I})^{-1}\mathbf{y}$$



What are the differences between hard and soft?

Not much different in practice.

Provable generalization guarantees for the soft one.

SSL with Graphs: Regularized Harmonic Functions

Larger implications of random walks

random walk relates to **commute distance** which should satisfy

($\star$) Vertices in the **same** cluster of the graph have a **small** commute distance, whereas two vertices in **different** clusters of the graph have a **large** commute distance.

Do we have this property for HS? What if $N \to \infty$?

Luxburg/Radl/Hein: *Getting lost in space: Large sample analysis of the commute distance* http://www.informatik.uni-hamburg.de/ML/contents/people/luxburg/publications/LuxburgRadlHein2010_PaperAndSupplement.pdf

Solutions? 1) γ_g 2) amplified commute distance 3) $\mathbf{L}^p$ 4) $\mathbf{L}^\star$ …

The goal of these solutions: **make them remember!**

SSL with Graphs: Out of sample extension

Both **MinCut** and **HFS** only inferred the labels on unlabeled data.

They are **transductive**.

What if a new point $\mathbf{x}_{n_l+n_u+1}$ arrives? also called out-of-sample extension

Option 1) Add it to the graph and recompute HFS.

Option 2) Make the algorithms **inductive**!

Allow to be defined everywhere: $f : \mathcal{X} \mapsto \mathbb{R}$
Allow $f(\mathbf{x}_i) \neq y_i$. Why? To deal with noise.

Solution: **Manifold Regularization**

SSL with Graphs: Manifold Regularization

General (S)SL objective:

$$\min_f \sum_i^{n_l} V(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \lambda\Omega(f)$$

Want to control f, also for the out-of-sample data, i.e., **everywhere**.

$$\Omega(f) = \lambda_2 \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f} + \lambda_1 \int_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})^2 \,\mathrm{d}\mathbf{x}$$

For general **kernels**:

$$\min_{f\in\mathcal{H}_\mathcal{K}} \sum_i^{n_l} V(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) + \lambda_1 \|f\|_\mathcal{K}^2 + \lambda_2 \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}$$

SSL with Graphs: Manifold Regularization

$$f^\star = \arg\min_{f \in \mathcal{H}_\mathcal{K}} \sum_{i}^{n_l} V(\mathbf{x}_i, y_i, f) + \lambda_1 \|f\|_\mathcal{K}^2 + \lambda_2 \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}$$

Representer Theorem for Manifold Regularization

The minimizer $f^\star$ has a **finite** expansion of the form

$$f^\star(\mathbf{x}) = \sum_{i=1}^{n_l+n_u} \alpha_i \mathcal{K}(\mathbf{x}, \mathbf{x}_i)$$

$$V(\mathbf{x}, y, f) = (y - f(\mathbf{x}))^2$$

LapRLS Laplacian Regularized Least Squares

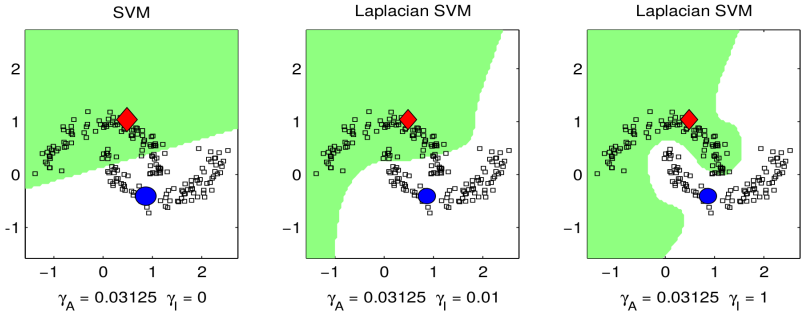
$$V(\mathbf{x}, y, f) = \max(0, 1 - yf(\mathbf{x}))$$

LapSVM Laplacian Support Vector Machines

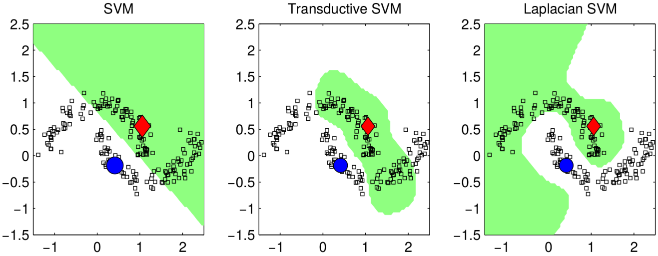
SSL with Graphs: Laplacian SVMs

$$f^\star = \arg\min_{f \in \mathcal{H}_\mathcal{K}} \sum_i^{n_l} \max\left(0, 1 - yf\left(\mathbf{x}\right)\right) + \gamma_A \|f\|_\mathcal{K}^2 + \gamma_I \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}$$

Allows us to learn a function in **RKHS**, i.e., **RBF** kernels.



SSL with Graphs: Laplacian SVMs



Checkpoint 1

Semi-supervised learning with graphs:

$$\min_{\mathbf{f}\in\{\pm 1\}^{n_l+n_u}} (\infty)\sum_{i=1}^{n_l} w_{ij}\left(f(\mathbf{x}_i)-y_i\right)^2+\lambda\sum_{i,j=1}^{n_l+n_u}\left(f(\mathbf{x}_i)-f(\mathbf{x}_j)\right)^2$$

Regularized harmonic Solution:

$$\mathbf{f}_u=\left(\mathbf{L}_{uu}+\gamma_g\mathbf{I}\right)^{-1}\left(\mathbf{W}_{ul}\mathbf{f}_l\right)$$

Checkpoint 2

Unconstrained regularization in general:

$$\mathbf{f}^\star = \min_{\mathbf{f}\in\mathbb{R}^N} (\mathbf{f}-\mathbf{y})^\mathsf{T}\mathbf{C}(\mathbf{f}-\mathbf{y}) + \mathbf{f}^\mathsf{T}\mathbf{Q}\mathbf{f}$$

Out of sample extension: Laplacian SVMs

$$f^\star = \arg\min_{f\in\mathcal{H}_\mathcal{K}} \sum_i^{n_l} \max\left(0, 1 - yf\left(\mathbf{x}\right)\right) + \lambda_1 \|f\|_\mathcal{K}^2 + \lambda_2 \mathbf{f}^\mathsf{T}\mathbf{L}\mathbf{f}$$

Michal Valko
michal.valko@inria.fr
ENS Paris-Saclay, MVA 2017/2018
SequeL team, Inria Lille — Nord Europe
https://team.inria.fr/sequel/