Reassuring and Troubling Views on Graph-Based Semi-Supervised Learning

Yoshua Bengio, Olivier Delalleau and Nicolas Le Roux

Université de Montréal

ICML Workshop - August 7th, 2005

Outline

1 Graph-Based Semi-Supervised Learning

2 Transduction, Induction, Approximation

3 Curse of Dimensionality

Notations

- Task of **binary classification** with labels $y_i \in \{-1, 1\}$

Notations

- Task of **binary classification** with labels $y_i \in \{-1, 1\}$
- $n = l + u$ labeled and unlabeled samples

Notations

- Task of **binary classification** with labels $y_i \in \{-1, 1\}$
- $n = l + u$ labeled and unlabeled samples
- Input part for **all** samples: $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$

Notations

- Task of **binary classification** with labels $y_i \in \{-1, 1\}$
- $n = l + u$ labeled and unlabeled samples
- Input part for **all** samples: $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$
- Target part for **labeled** samples: $\boldsymbol{Y}_l = (y_1, y_2, \ldots, y_l)^\top$

Notations

- Task of **binary classification** with labels $y_i \in \{-1, 1\}$
- $n = l + u$ labeled and unlabeled samples
- Input part for **all** samples: $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$
- Target part for **labeled** samples: $\boldsymbol{Y}_l = (y_1, y_2, \ldots, y_l)^\top$
- **Transduction** $\Rightarrow \hat{\boldsymbol{Y}}_u = (\hat{y}_{l+1}, \hat{y}_{l+2}, \ldots, \hat{y}_n)^\top$

Notations

- Task of **binary classification** with labels $y_i \in \{-1, 1\}$
- $n = l + u$ labeled and unlabeled samples
- Input part for **all** samples: $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$
- Target part for **labeled** samples: $\boldsymbol{Y}_l = (y_1, y_2, \ldots, y_l)^\top$
- **Transduction** $\Rightarrow \hat{\boldsymbol{Y}}_u = (\hat{y}_{l+1}, \hat{y}_{l+2}, \ldots, \hat{y}_n)^\top$
- **Induction** $\Rightarrow \hat{y} : \boldsymbol{x} \rightarrow \hat{y}(\boldsymbol{x})$

Cost Criterion for Semi-Supervised Learning

A good estimated labeling $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_l, \hat{\boldsymbol{Y}}_u)$ should be:

Cost Criterion for Semi-Supervised Learning

A good estimated labeling $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_l, \hat{\boldsymbol{Y}}_u)$ should be:

1. consistent with the given labels: $\hat{\boldsymbol{Y}}_l \simeq \boldsymbol{Y}_l$

Cost Criterion for Semi-Supervised Learning

A good estimated labeling $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_l, \hat{\boldsymbol{Y}}_u)$ should be:

1. consistent with the given labels: $\hat{\boldsymbol{Y}}_l \simeq \boldsymbol{Y}_l$
2. smooth on the manifold where the data lie (*manifold assumption*):

$$\hat{y}_i \simeq \hat{y}_j \text{ when } \boldsymbol{x}_i \text{ close to } \boldsymbol{x}_j$$

Cost Criterion for Semi-Supervised Learning

A good estimated labeling $\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_l, \hat{\boldsymbol{Y}}_u)$ should be:

1. consistent with the given labels: $\hat{\boldsymbol{Y}}_l \simeq \boldsymbol{Y}_l$
2. smooth on the manifold where the data lie (*manifold assumption*):

$$\hat{y}_i \simeq \hat{y}_j \text{ when } \boldsymbol{x}_i \text{ close to } \boldsymbol{x}_j$$

Trade-off between (1) and (2) $\Rightarrow$ cost function:

$$C(\hat{\boldsymbol{Y}}) = \sum_{i=1}^{l} (\hat{y}_i - y_i)^2 + \frac{\mu}{2} \sum_{i,j=1}^{n} \boldsymbol{W}_{ij} (\hat{y}_i - \hat{y}_j)^2$$

with $\boldsymbol{W}_{ij} = W_X(\boldsymbol{x}_i, \boldsymbol{x}_j)$ a positive weighting function (e.g kernel)

Graph-Based Cost Criterion

Graph Laplacian: $\boldsymbol{L}_{ii} = \sum_{j \neq i} \boldsymbol{W}_{ij}$ and $\boldsymbol{L}_{ij} = -\boldsymbol{W}_{ij}$.

$$\begin{aligned} C(\hat{\boldsymbol{Y}}) &= \sum_{i=1}^{l} (\hat{y}_i - y_i)^2 + \frac{\mu}{2} \sum_{i,j=1}^{n} \boldsymbol{W}_{ij} (\hat{y}_i - \hat{y}_j)^2 \\ &= \|\hat{\boldsymbol{Y}}_l - \boldsymbol{Y}_l\|^2 + \mu \hat{\boldsymbol{Y}}^\top \boldsymbol{L} \hat{\boldsymbol{Y}} \end{aligned}$$

Graph-Based Cost Criterion

Graph Laplacian: $\boldsymbol{L}_{ii} = \sum_{j \neq i} \boldsymbol{W}_{ij}$ and $\boldsymbol{L}_{ij} = -\boldsymbol{W}_{ij}$.

$$\begin{aligned} C(\hat{\boldsymbol{Y}}) &= \sum_{i=1}^{l} (\hat{y}_i - y_i)^2 + \frac{\mu}{2} \sum_{i,j=1}^{n} \boldsymbol{W}_{ij} (\hat{y}_i - \hat{y}_j)^2 \\ &= \|\hat{\boldsymbol{Y}}_l - \boldsymbol{Y}_l\|^2 + \mu \hat{\boldsymbol{Y}}^\top \boldsymbol{L} \hat{\boldsymbol{Y}} \end{aligned}$$

$C(\hat{\boldsymbol{Y}})$ is minimized when

$$(\boldsymbol{S} + \mu \boldsymbol{L}) \hat{\boldsymbol{Y}} = \boldsymbol{Y}$$

with $\boldsymbol{S}_{ij} = \delta_{i=j} \delta_{i \leq l}$
$\Rightarrow$ **linear system (*n* unknowns and equations)**

From Matrix Inversion to Label Propagation

Linear system rewrites for a labeled point

$$\hat{y}_i = \frac{\sum_j \boldsymbol{W}_{ij}\hat{y}_j + \frac{1}{\mu}y_i}{\sum_j \boldsymbol{W}_{ij} + \frac{1}{\mu}}$$

and for an unlabeled point

$$\hat{y}_i = \frac{\sum_j \boldsymbol{W}_{ij}\hat{y}_j}{\sum_j \boldsymbol{W}_{ij}}.$$

From Matrix Inversion to Label Propagation

Linear system rewrites for a labeled point

$$\hat{y}_i^{(t+1)} = \frac{\sum_j \boldsymbol{W}_{ij} \hat{y}_j^{(t)} + \frac{1}{\mu} y_i}{\sum_j \boldsymbol{W}_{ij} + \frac{1}{\mu}}$$

and for an unlabeled point

$$\hat{y}_i^{(t+1)} = \frac{\sum_j \boldsymbol{W}_{ij} \hat{y}_j^{(t)}}{\sum_j \boldsymbol{W}_{ij}}.$$

$\Rightarrow$ **Jacobi** or **Gauss-Seidel** iterative algorithms (but there are more refined algorithms for sparse system resolution)

Related Work and Variants

- M. Szummer and T. Jaakkola (2002): *Partially labeled classification with Markov random walks*
- X. Zhu, Z. Ghahramani and J. Lafferty (2003): *Semi-supervised learning using Gaussian fields and harmonic functions*
- D. Zhou, O. Bousquet, T. Navin Lal, J. Weston, B. Schölkopf (2004): *Learning with local and global consistency*
- M. Belkin, I. Matveeva and P. Niyogi (2004): *Regularization and Semi-supervised Learning on Large Graphs*
- . . .

Analogies

Analogies

- **Electric networks** (Doyle and Snell, 1984; Zhu, Ghahramani and Lafferty, 2003): the estimated label at a node is the same as its potential in an electric network where resistors between nodes are such that $\boldsymbol{R}_{ij} = \boldsymbol{W}_{ij}^{-1}$, negative labels are linked to a $-1V$ generator, and positive labels to a $+1V$ generator.

Analogies

- **Electric networks** (Doyle and Snell, 1984; Zhu, Ghahramani and Lafferty, 2003): the estimated label at a node is the same as its potential in an electric network where resistors between nodes are such that $\boldsymbol{R}_{ij} = \boldsymbol{W}_{ij}^{-1}$, negative labels are linked to a $-1V$ generator, and positive labels to a $+1V$ generator.
- **Markov random walks** (Szummer and Jaakkola, 2002; Zhu, Ghahramani and Lafferty, 2003): with labels 0 and 1 (instead of 0 and -1), the estimated label at a node is equal to the probability of ending at a sample with label 1 when starting from the node and performing a random walk with transition probabilities proportional to the weights $\boldsymbol{W}_{ij}$.

Analogies

- **Electric networks** (Doyle and Snell, 1984; Zhu, Ghahramani and Lafferty, 2003): the estimated label at a node is the same as its potential in an electric network where resistors between nodes are such that $\boldsymbol{R}_{ij} = \boldsymbol{W}_{ij}^{-1}$, negative labels are linked to a $-1V$ generator, and positive labels to a $+1V$ generator.
- **Markov random walks** (Szummer and Jaakkola, 2002; Zhu, Ghahramani and Lafferty, 2003): with labels 0 and 1 (instead of 0 and -1), the estimated label at a node is equal to the probability of ending at a sample with label 1 when starting from the node and performing a random walk with transition probabilities proportional to the weights $\boldsymbol{W}_{ij}$.
- **Heat diffusion**, ...

Why Induction?

The previously presented techniques perform **transduction**.

Out-of-sample predictions

What to do with a new test point?

We could retrain, but computationally expensive $O(n^3)$ (or $O((kn)^{3/2})$ if sparse graph with k neighbors).

Trade-off

If we could do induction cheaply, even at the price of losing a bit of the advantage of transduction, it might be useful in practice!

Induction Criterion in a Transductive Framework

Induction as an **approximation to transduction**: force predictor's response to remain fixed on previous training data (labeled and unlabeled), when test point is added.

From Transduction to Induction

Solving the linear system $\Rightarrow$ $\hat{\boldsymbol{Y}}$ (**transduction**).
From a new point $\boldsymbol{x}$ and *already computed* $\hat{\boldsymbol{Y}}$:

$$\min_{\hat{y}(\boldsymbol{x})} C(\hat{y}(\boldsymbol{x})) = C(\hat{\boldsymbol{Y}}) + \frac{\mu}{2} \sum_{i=1}^{n} W_X(\boldsymbol{x}_i, \boldsymbol{x})(\hat{y}_i - \hat{y}(\boldsymbol{x}))^2$$

From Transduction to Induction

Solving the linear system $\Rightarrow$ $\hat{\boldsymbol{Y}}$ (**transduction**).
From a new point $\boldsymbol{x}$ and *already computed* $\hat{\boldsymbol{Y}}$:

$$\min_{\hat{y}(\boldsymbol{x})} C(\hat{y}(\boldsymbol{x})) = C(\hat{\boldsymbol{Y}}) + \frac{\mu}{2} \sum_{i=1}^{n} W_X(\boldsymbol{x}_i, \boldsymbol{x})(\hat{y}_i - \hat{y}(\boldsymbol{x}))^2$$

$$\Rightarrow \hat{y}(\boldsymbol{x}) = \frac{\sum_{i=1}^{n} W_X(\boldsymbol{x}_i, \boldsymbol{x})\hat{y}_i}{\sum_{i=1}^{n} W_X(\boldsymbol{x}_i, \boldsymbol{x})}$$

Induction like Parzen Windows, but using *estimated* labels $\hat{\boldsymbol{Y}}$

Faster (Approximate) Training from Subset

- Previous algorithms are at least quadratic in n.

Faster (Approximate) Training from Subset

- Previous algorithms are at least quadratic in n.
- Induction formula $\Rightarrow$ could train only on subset S containing the labeled samples and **some** unlabeled ones. For $\boldsymbol{x}_i \in R = X \setminus S$:

$$\hat{y}_i \simeq \frac{\sum_{j \in S} \boldsymbol{W}_{ij} \hat{y}_j}{\sum_{j \in S} \boldsymbol{W}_{ij}}$$

i.e. $\hat{\boldsymbol{Y}}_R \simeq \overline{\boldsymbol{W}}_{RS} \hat{\boldsymbol{Y}}_S$.

Faster (Approximate) Training from Subset

- Previous algorithms are at least quadratic in n.
- Induction formula $\Rightarrow$ could train only on subset S containing the labeled samples and **some** unlabeled ones. For $\boldsymbol{x}_i \in R = X \setminus S$:

$$\hat{y}_i \simeq \frac{\sum_{j \in S} \boldsymbol{W}_{ij} \hat{y}_j}{\sum_{j \in S} \boldsymbol{W}_{ij}}$$

  i.e. $\hat{\boldsymbol{Y}}_R \simeq \overline{\boldsymbol{W}}_{RS} \hat{\boldsymbol{Y}}_S$.
- Better: minimize **the full cost** over $\hat{\boldsymbol{Y}}_S$ only. $C(\hat{\boldsymbol{Y}}) = C(\hat{\boldsymbol{Y}}_S, \hat{\boldsymbol{Y}}_R) \simeq C(\hat{\boldsymbol{Y}}_S, \overline{\boldsymbol{W}}_{RS} \hat{\boldsymbol{Y}}_S) = C'(\hat{\boldsymbol{Y}}_S)$

$\Rightarrow$ linear system with only $|S|$ unknowns.

More Approximations

$$
\begin{aligned}
C'(\hat{\boldsymbol{Y}}_S) = & \underbrace{\|\hat{\boldsymbol{Y}}_I - \boldsymbol{Y}_I\|^2}_{C_L} && O(l) \\
& + \underbrace{\mu \hat{\boldsymbol{Y}}_S{}^\top \boldsymbol{L}_{SS} \hat{\boldsymbol{Y}}_S}_{C_{SS}} && O(|S|^2) \\
& + \underbrace{2\mu \hat{\boldsymbol{Y}}_R{}^\top \boldsymbol{L}_{RS} \hat{\boldsymbol{Y}}_S}_{C_{RS}} && O(|S||R|) \\
& + \underbrace{\mu \hat{\boldsymbol{Y}}_R{}^\top \boldsymbol{L}_{RR} \hat{\boldsymbol{Y}}_R}_{C_{RR}} && O(|R|^2)
\end{aligned}
$$

Cost computation complexity: $O(|R|^2)$ is too much!

More Approximations

$$
\begin{aligned}
C'(\hat{\boldsymbol{Y}}_S) = & \underbrace{\|\hat{\boldsymbol{Y}}_I - \boldsymbol{Y}_I\|^2}_{C_L} & O(l) \\
& + \underbrace{\mu \hat{\boldsymbol{Y}}_S{}^\top \boldsymbol{L}_{SS} \hat{\boldsymbol{Y}}_S}_{C_{SS}} & O(|S|^2) \\
& + \underbrace{2\mu \hat{\boldsymbol{Y}}_R{}^\top \boldsymbol{L}_{RS} \hat{\boldsymbol{Y}}_S}_{C_{RS}} & O(|S||R|)
\end{aligned}
$$

Cost computation complexity: $O(|S||R|)$ is ok!

Justifications for the Proposed Cost

- Keeping C_{RR} would lead to a useless algorithm (too slow)

Justifications for the Proposed Cost

- Keeping C_{RR} would lead to a useless algorithm (too slow)
- Removing C_{RS} would mean training only on S (too simple)

Justifications for the Proposed Cost

- Keeping C_{RR} would lead to a useless algorithm (too slow)
- Removing C_{RS} would mean training only on S (too simple)
- Thus we have no choice but to use the proposed cost

Justifications for the Proposed Cost

- Keeping C_{RR} would lead to a useless algorithm (too slow)
- Removing C_{RS} would mean training only on S (too simple)
- Thus we have no choice but to use the proposed cost
- Experiments show it is often better! The approximation

$$\hat{y}_i \simeq \frac{\sum_{k \in S} \boldsymbol{W}_{ik} \hat{y}_k}{\sum_{k \in S} \boldsymbol{W}_{ik}}$$

can be very poor if $\boldsymbol{x}_i \in R$ is far from all samples $\boldsymbol{x}_k \in S$, which can lead to irrelevant terms in C_{RR} *with a significant weight* $\Rightarrow$ overall accuracy is worse

Subset Selection

1. **Random**:

Subset Selection

1. **Random**: fast,

Subset Selection

1. **Random**: fast, easy,

Subset Selection

1. **Random**: fast, easy, crappy.
   Main problem = does not “fill the space” well enough $\Rightarrow$ bad approximation by the induction formula (some points have no near neighbors in the subset)

Subset Selection

1. **Random**: fast, easy, crappy.
   Main problem = does not "fill the space" well enough $\Rightarrow$ bad approximation by the induction formula (some points have no near neighbors in the subset)
2. **Heuristic**: greedy construction of subset. Start with the labeled points ($S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\}$) and iteratively add

$$\boldsymbol{x}_i^* = \text{argmax}_{\boldsymbol{x}_i}\ dist(\boldsymbol{x}_i, S) = \text{argmin}_{\boldsymbol{x}_i} \sum_{j \in S} \boldsymbol{W}_{ij}$$

   i.e. choose the point $\boldsymbol{x}_i$ "farthest" from the current subset. (NB: additional tricks to eliminate outliers and sample more points near decision surface)

Experimental Results

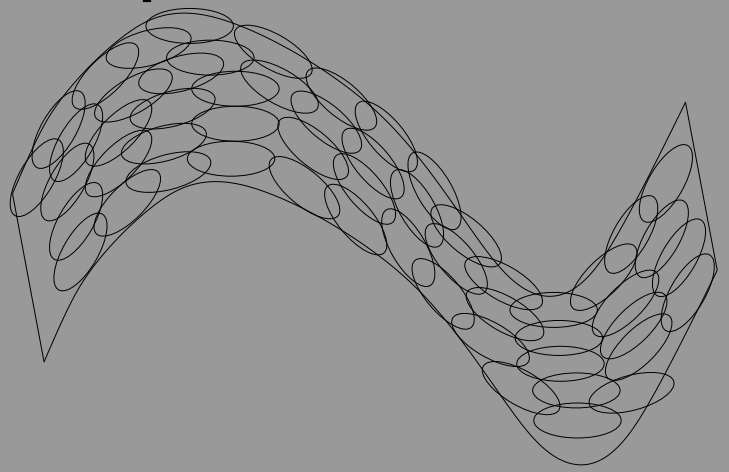
Table: Comparative Classification Error (Induction)

% labeled	LETTERS	MNIST	COVTYPE
1%			
NoSub	56.0	35.8	47.3
$RandSub_{subOnly}$	59.8	29.6	**44.8**
RandSub	57.4	27.7	75.7
SmartSub	**55.8**	**24.4**	45.0
5%			
NoSub	**27.1**	12.8	37.1
$RandSub_{subOnly}$	32.1	14.9	**35.4**
RandSub	29.1	12.6	70.6
SmartSub	28.5	**12.3**	35.8
10%			
NoSub	**18.8**	**9.5**	34.7
$RandSub_{subOnly}$	22.5	11.4	**32.4**
RandSub	20.3	9.7	64.7
SmartSub	19.8	**9.5**	33.4

More comparisons between *RandSub* and *SmartSub* on 8 more UCI datasets $\Rightarrow$ *SmartSub* always performs better.

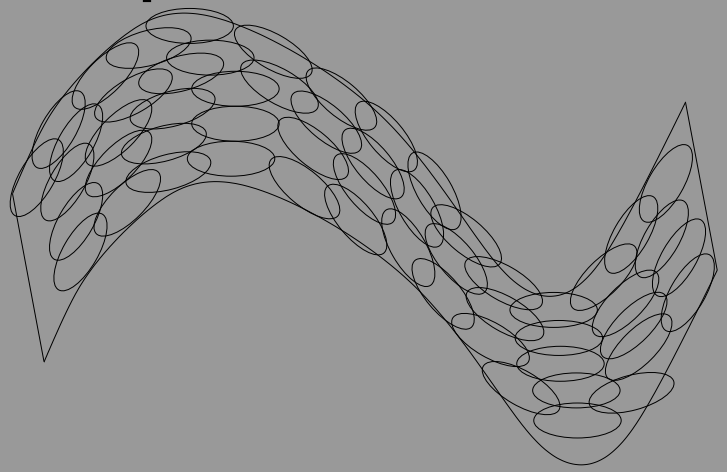
Curse of Dimensionality: Geometric Intuition

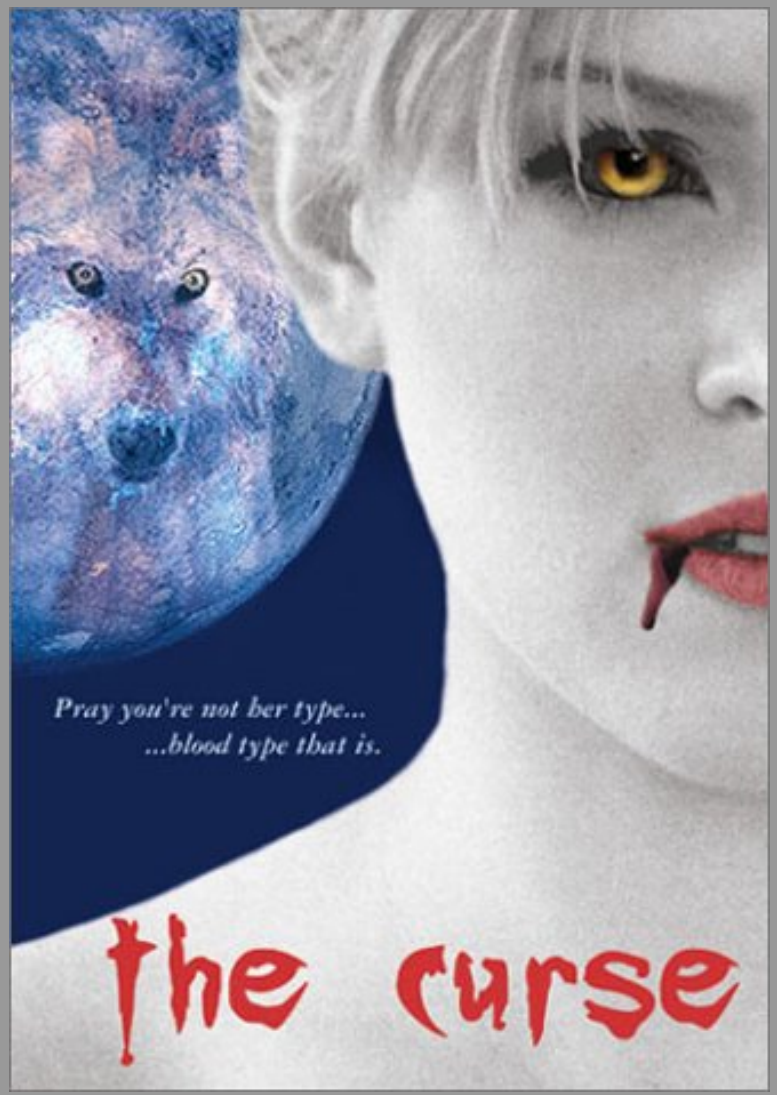
If we have to tile the space or manifold where bulk of the distribution is concentrated, then will need **exponential number of "patches"**:



Curse of Dimensionality: Geometric Intuition

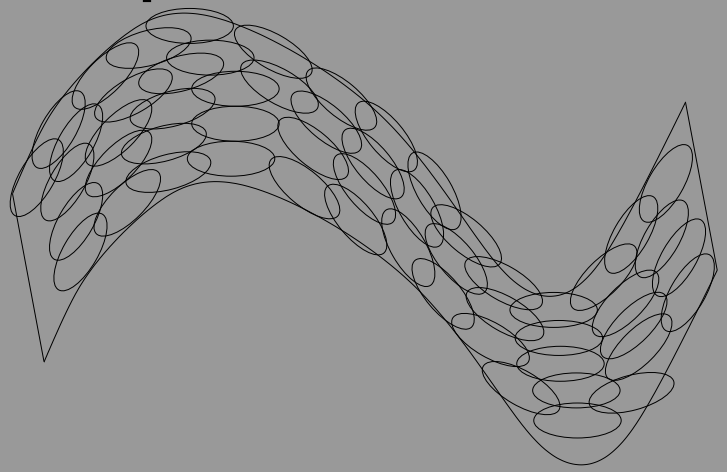
If we have to tile the space or manifold where bulk of the distribution is concentrated, then will need **exponential number of "patches"**:



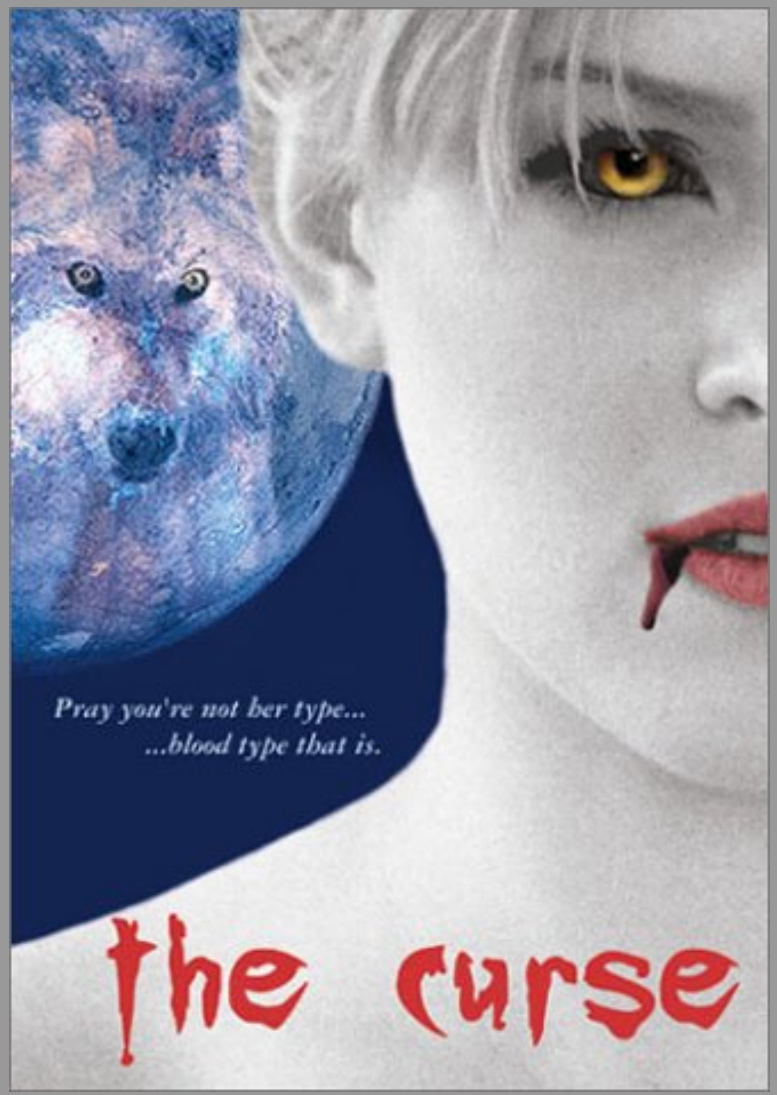


Curse of Dimensionality: Geometric Intuition

If we have to tile the space or manifold where bulk of the distribution is concentrated, then will need **exponential number of "patches"**:

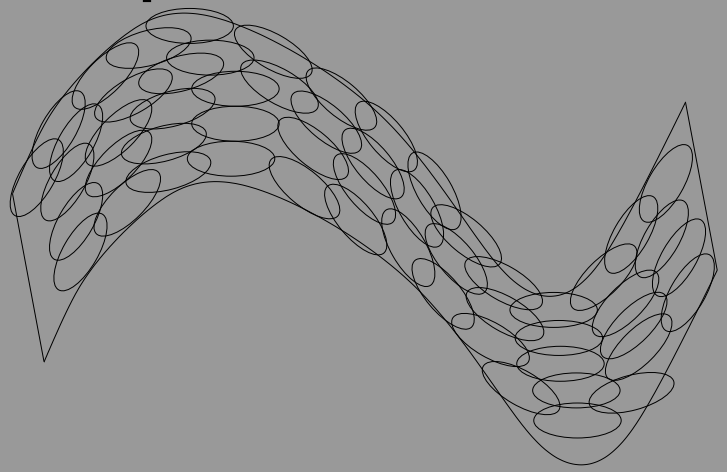


For classification, no need to cover the whole space/manifold, only decision surface, but still has dim. $d - 1$.



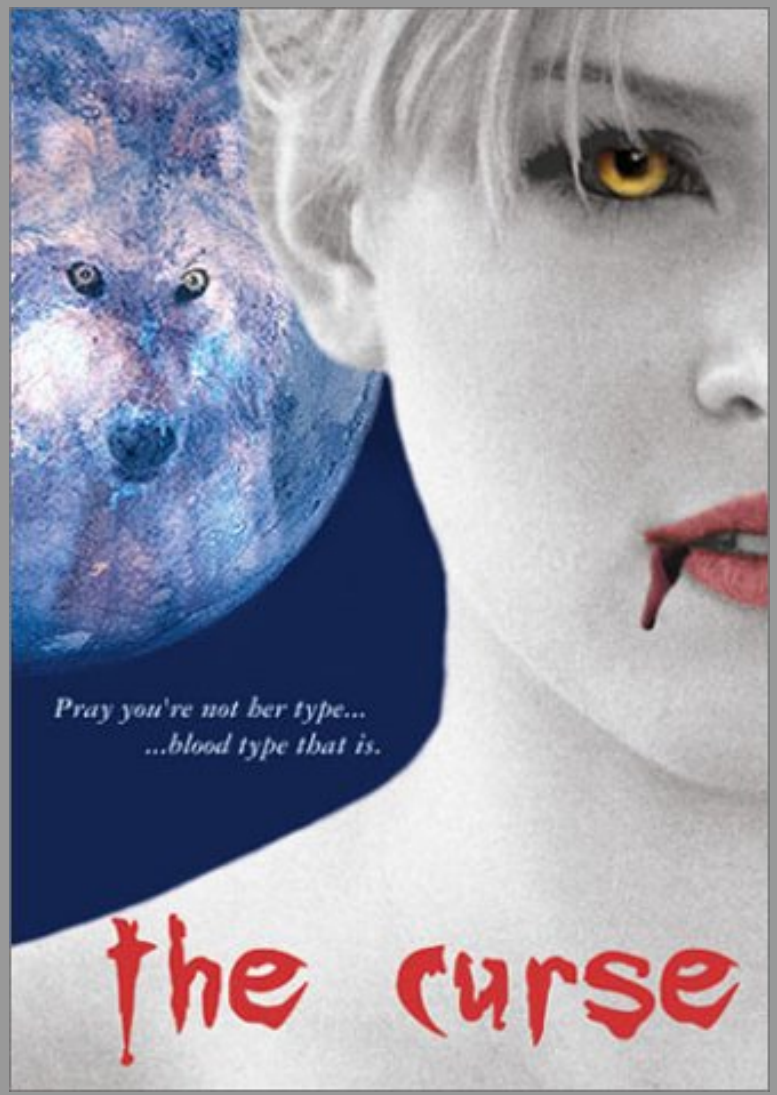
Curse of Dimensionality: Geometric Intuition

If we have to tile the space or manifold where bulk of the distribution is concentrated, then will need **exponential number of "patches"**:



For classification, no need to cover the whole space/manifold, only decision surface, but still has dim. $d - 1$.

May require $O(\text{const}^d)$ examples!



Curse of Dimensionality (1)

Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$

Curse of Dimensionality (1)

Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$

Curse #1 With W_X a local similarity function the decision surface is highly constrained $\Rightarrow$ may need lots of **unlabeled** data to be faithfully represented.

Curse of Dimensionality (1)

Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$

Curse #1 With W_X a local similarity function the decision surface is highly constrained $\Rightarrow$ may need lots of **unlabeled** data to be faithfully represented.

General argument (also applies to local kernel machines such as SVMs):

Curse of Dimensionality (1)

Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$

Curse #1 With W_X a local similarity function the decision surface is highly constrained $\Rightarrow$ may need lots of **unlabeled** data to be faithfully represented.

General argument (also applies to local kernel machines such as SVMs):

1. for any point $\boldsymbol{x}$ on the (estimated) decision surface, a smoothness property P is verified in neighborhood $\mathcal{N}(\boldsymbol{x})$ (e.g. with Gaussian kernel, the normal vector is constrained in a ball of radius proportional to σ)

Curse of Dimensionality (1)

Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$
Curse #1 With W_X a local similarity function the decision surface is highly constrained $\Rightarrow$ may need lots of **unlabeled** data to be faithfully represented.
General argument (also applies to local kernel machines such as SVMs):

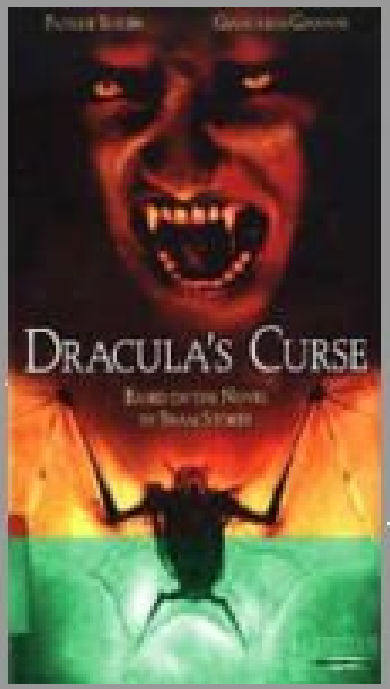
1. for any point $\boldsymbol{x}$ on the (estimated) decision surface, a smoothness property P is verified in neighborhood $\mathcal{N}(\boldsymbol{x})$ (e.g. with Gaussian kernel, the normal vector is constrained in a ball of radius proportional to σ)
2. in order to approximate the (true) decision surface correctly, $\mathcal{N}(\boldsymbol{x})$ must be small enough, as P is only valid locally (e.g. the true normal vector may vary a lot)

Curse of Dimensionality (1)

Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$
Curse #1 With W_X a local similarity function the decision surface is highly constrained $\Rightarrow$ may need lots of **unlabeled** data to be faithfully represented.
General argument (also applies to local kernel machines such as SVMs):

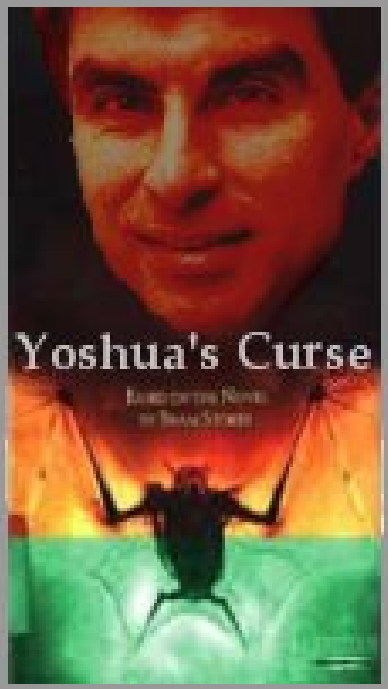
1. for any point $\boldsymbol{x}$ on the (estimated) decision surface, a smoothness property P is verified in neighborhood $\mathcal{N}(\boldsymbol{x})$ (e.g. with Gaussian kernel, the normal vector is constrained in a ball of radius proportional to σ)
2. in order to approximate the (true) decision surface correctly, $\mathcal{N}(\boldsymbol{x})$ must be small enough, as P is only valid locally (e.g. the true normal vector may vary a lot)
3. for $\mathcal{N}(\boldsymbol{x})$ to be small, one needs many training samples (possibly a number exponential in the dimension)

Curse of Dimensionality (2)



Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$

Curse of Dimensionality (2)



Labeling function: $\hat{y}(\boldsymbol{x}) = \sum_i \hat{y}_i \overline{W}_X(\boldsymbol{x}_i, \boldsymbol{x})$

Curse #2 The number of labeled samples must be higher than the number of regions with constant label

$\Rightarrow$ may need lots of **labeled** data if there are many such regions (possibly exponential in the dimension)

Highly Varying: Parity Function

parity:

$$(b_1, \ldots, b_d) \in \{0,1\}^d \mapsto \begin{cases} 1 \text{ if even } \sum_{i=1}^{d} b_i \\ -1 \text{ otherwise} \end{cases}$$

Highly Varying: Parity Function

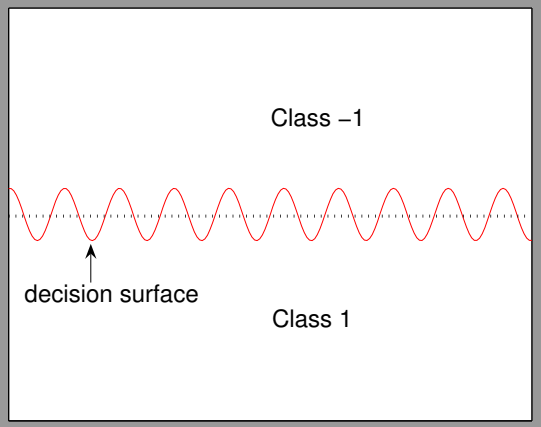
parity:

$$(b_1, \ldots, b_d) \in \{0,1\}^d \mapsto \begin{cases} 1 \text{ if even } \sum_{i=1}^{d} b_i \\ -1 \text{ otherwise} \end{cases}$$

Theorem

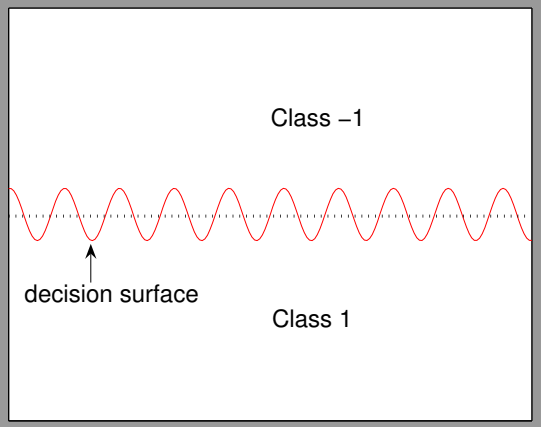
A Gaussian kernel classifier needs at least 2^{d-1} Gaussians (i.e. support vectors) to learn the parity function (when Gaussians have fixed width and are centered on training points).

Simple but Highly Variable Functions: Difficult to Learn



This “complex” sinusoidal decision surface requires many Gaussians to learn, but in “C” language, it has a high prior.
***** KEY RESULT *****

Simple but Highly Variable Functions: Difficult to Learn



This “complex” sinusoidal decision surface requires many Gaussians to learn, but in “C” language, it has a high prior.
***** KEY RESULT *****

Corollary of (Schmitt 2002)

If $\exists$ a line in $\boldsymbol{R}^d$ that intersects m times with the decision surface S (and is not included in S), then one needs at least $\lceil \frac{m}{2} \rceil$ Gaussians (of same width) to learn S with a Gaussian kernel classifier.

Local-Derivative Kernels

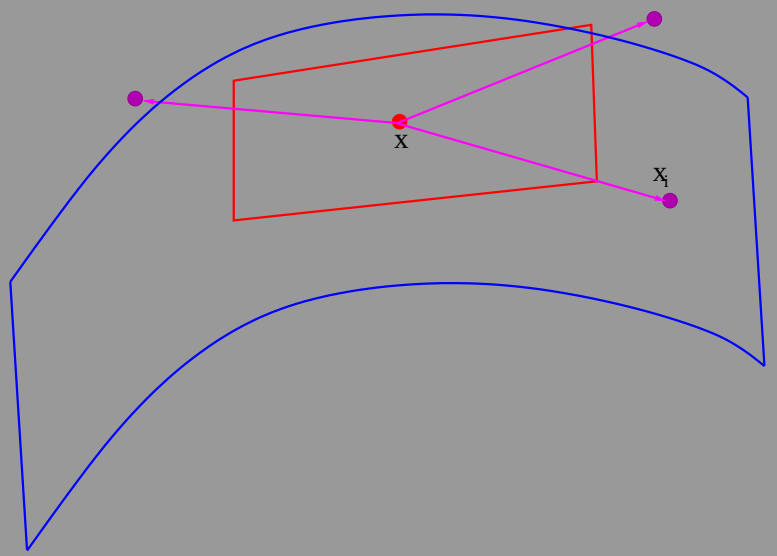
The derivative of kernel predictor f is $\frac{\partial f(x)}{\partial x} = \sum_{i=1}^{n} \alpha_i \frac{\partial K(x, x_i)}{\partial x}$.

Local-Derivative Kernels

The derivative of kernel predictor f is $\frac{\partial f(x)}{\partial x} = \sum_{i=1}^{n} \alpha_i \frac{\partial K(x, x_i)}{\partial x}$.

Local-derivative kernel: when $\partial f / \partial x$ is (approximately) contained in **the span of the vectors** $(x - x_j)$ **with** x_j **a neighbor of** x:

$$\frac{\partial f(x)}{\partial x} \simeq \sum_{x_j \in \mathcal{N}(x)} \gamma_j (x - x_j)$$

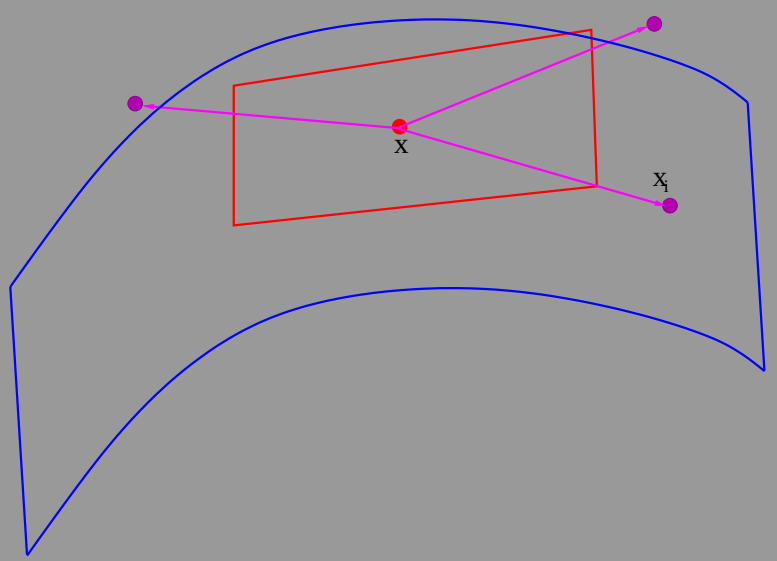


Local-Derivative Kernels

The derivative of kernel predictor f is $\frac{\partial f(x)}{\partial x} = \sum_{i=1}^{n} \alpha_i \frac{\partial K(x, x_i)}{\partial x}$.

Local-derivative kernel: when $\partial f / \partial x$ is (approximately) contained in **the span of the vectors $(x - x_j)$ with x_j a neighbor of x**:

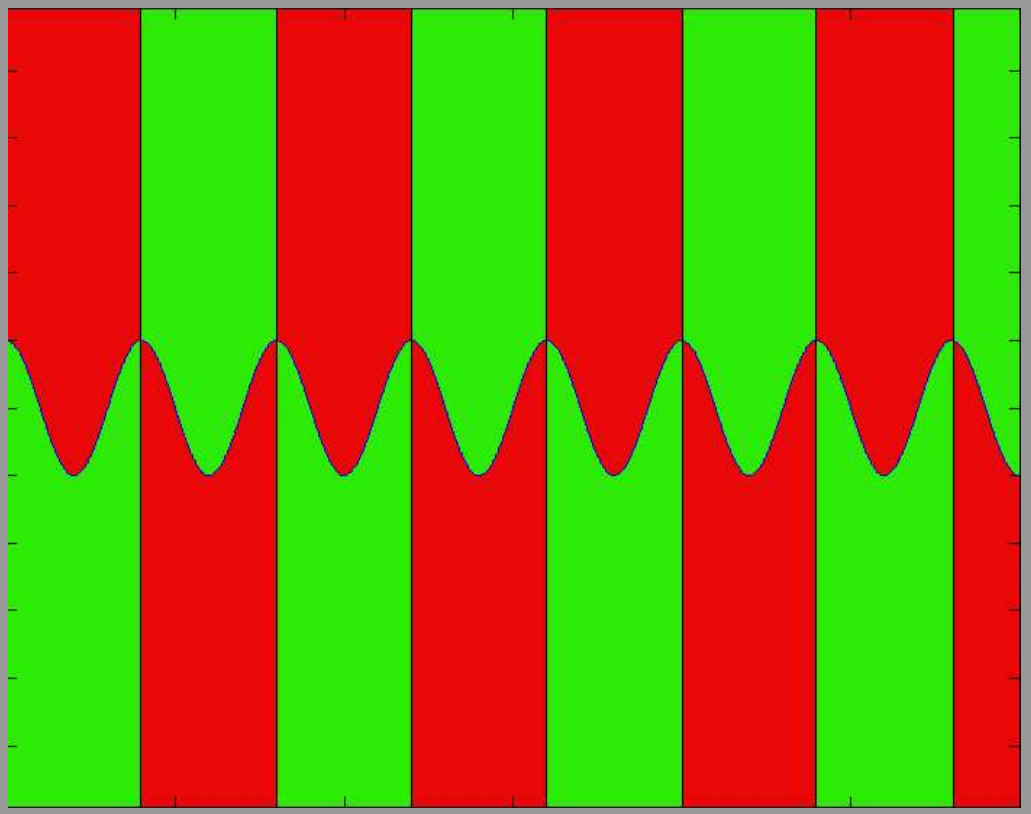
$$\frac{\partial f(x)}{\partial x} \simeq \sum_{x_j \in \mathcal{N}(x)} \gamma_j (x - x_j)$$



Bad News!

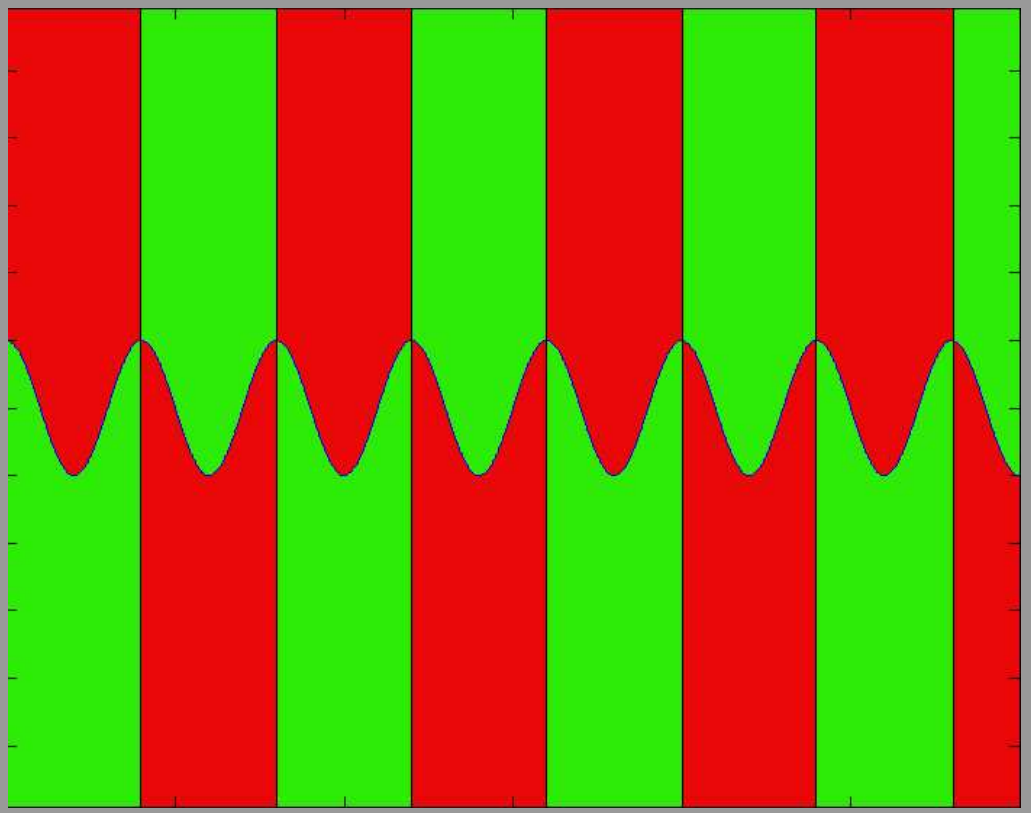
Shape of manifold (tangent vector at x) mostly determined by neighbors of x in the graph.

Semi-Supervised Need of Examples



Need unlabeled examples along *sinus* decision surface + labeled ones in each class region.

Semi-Supervised Need of Examples

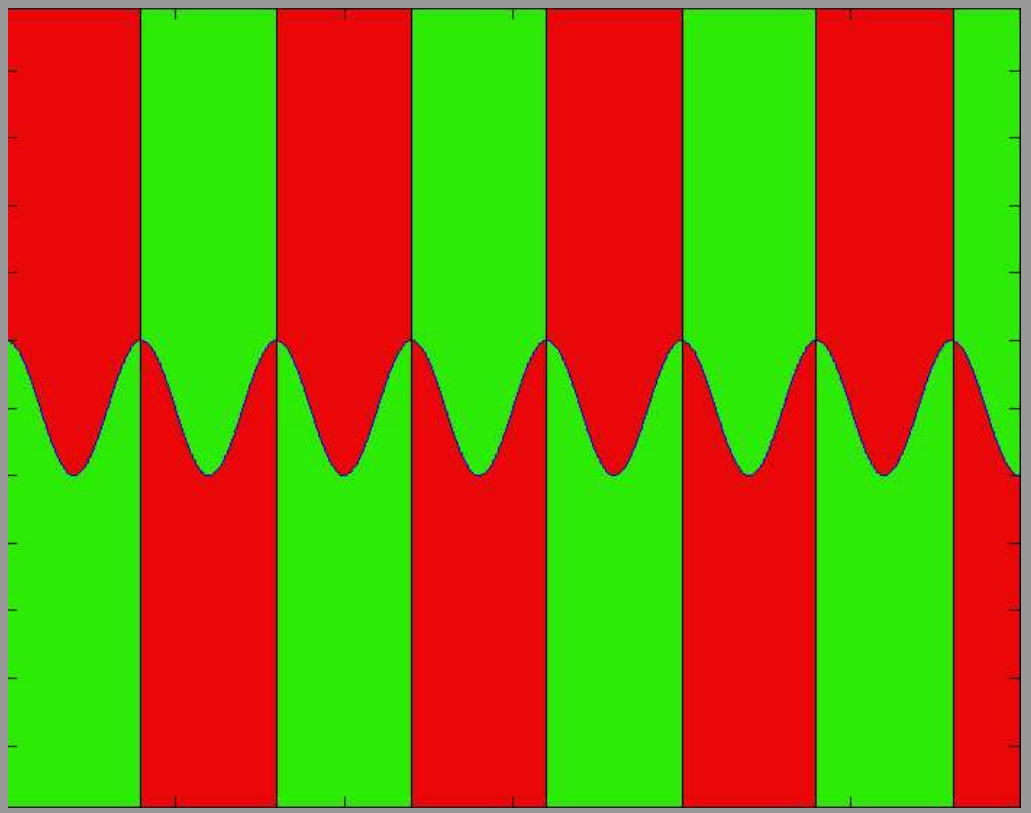


Need unlabeled examples along *sinus* decision surface + labeled ones in each class region.
Number of labeled and unlabeled examples to learn such decision surface may grow **exponentially** with dimension.

Theorem

After running a label propagation algorithm minimizing quadratic cost, number of regions with constant estimated label $\leq$ number of labeled examples.

Semi-Supervised Need of Examples



Need unlabeled examples along *sinus* decision surface + labeled ones in each class region.
Number of labeled and unlabeled examples to learn such decision surface may grow **exponentially** with dimension.
This could be **easily** learned with a non-local learning algorithm.

Theorem

After running a label propagation algorithm minimizing quadratic cost, number of regions with constant estimated label $\leq$ number of labeled examples.

Philosophical Question

Most common non-parametric approaches based on smoothness prior, which leads to “local” learning algorithms, e.g. most kernel-based ones, yielding to **curse of dimensionality** = difficulty to learn structured but highly variable functions.

Philosophical Question

Most common non-parametric approaches based on smoothness prior, which leads to “local” learning algorithms, e.g. most kernel-based ones, yielding to **curse of dimensionality** = difficulty to learn structured but highly variable functions.

No-free-lunch thm: no universal recipe without appropriate prior.

Philosophical Question

Most common non-parametric approaches based on smoothness prior, which leads to “local” learning algorithms, e.g. most kernel-based ones, yielding to **curse of dimensionality** = difficulty to learn structured but highly variable functions.

No-free-lunch thm: no universal recipe without appropriate prior.

Smoothness may not be the only way to obtain “simple functions”: e.g. According to Kolmogorov complexity, $sinus(x)$ and $parity(x)$ are simple yet they are highly variable (apparently complex) functions.

Philosophical Question

Most common non-parametric approaches based on smoothness prior, which leads to "local" learning algorithms, e.g. most kernel-based ones, yielding to **curse of dimensionality** = difficulty to learn structured but highly variable functions.

No-free-lunch thm: no universal recipe without appropriate prior.

Smoothness may not be the only way to obtain "simple functions": e.g. According to Kolmogorov complexity, $sinus(x)$ and $parity(x)$ are simple yet they are highly variable (apparently complex) functions.

Is there hope? Humans seem to learn highly-varying yet structured functions! There might be loose enough priors on general classes of functions that allow non-local learning algorithms to learn them.

Conclusion

Conclusion

- Simple non-parametric setting $\Rightarrow$ powerful non-parametric semi-supervised algorithm

Conclusion

- Simple non-parametric setting $\Rightarrow$ powerful non-parametric semi-supervised algorithm
- Can scale to large datasets thanks to sparsity / subset selection

Conclusion

- Simple non-parametric setting $\Rightarrow$ powerful non-parametric semi-supervised algorithm
- Can scale to large datasets thanks to sparsity / subset selection
- Interesting physical analogies

Conclusion

- Simple non-parametric setting $\Rightarrow$ powerful non-parametric semi-supervised algorithm
- Can scale to large datasets thanks to sparsity / subset selection
- Interesting physical analogies
- Limitations of local weights: curse of dimensionality

Conclusion

- Simple non-parametric setting $\Rightarrow$ powerful non-parametric semi-supervised algorithm
- Can scale to large datasets thanks to sparsity / subset selection
- Interesting physical analogies
- Limitations of local weights: curse of dimensionality
- Is there hope to discover **non-local** semi-supervised learning? yes!

References

Belkin, M., Matveeva, I., and Niyogi, P. (2004).
Regularization and semi-supervised learning on large graphs.
In Shawe-Taylor, J. and Singer, Y., editors, *COLT'2004*. Springer.

Delalleau, O., Bengio, Y., and Le Roux, N. (2005).
Efficient non-parametric function induction in semi-supervised learning.
In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.

Doyle, P. G. and Snell, J. L. (1984).
Random walks and electric networks.
Mathematical Association of America.

Szummer, M. and Jaakkola, T. (2002).
Partially labeled classification with markov random walks.
In Dietterich, T., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

Zhou, D., Bousquet, O., Navin Lal, T., Weston, J., and Schölkopf, B. (2004).
Learning with local and global consistency.
In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*, Cambridge, MA. MIT Press.

Zhu, X., Ghahramani, Z., and Lafferty, J. (2003).
Semi-supervised learning using Gaussian fields and harmonic functions.
In *ICML'2003*.