11 Label Propagation and Quadratic Criterion

Yoshua Bengio
Olivier Delalleau
Nicolas Le Roux

Various graph-based algorithms for semi-supervised learning have been proposed in the recent literature. They rely on the idea of building a graph whose nodes are data points (labeled and unlabeled) and edges represent similarities between points. Known labels are used to propagate information through the graph in order to label all nodes. In this chapter, we show how these different algorithms can be cast into a common framework where one minimizes a quadratic cost criterion whose closed-form solution is found by solving a linear system of size n (total number of data points). The cost criterion naturally leads to an extension of such algorithms to the inductive setting, where one obtains test samples one at a time: the derived induction formula can be evaluated in $O(n)$ time, which is much more efficient than solving again exactly the linear system (which in general costs $O(kn^2)$ time for a sparse graph where each data point has k neighbors). We also use this inductive formula to show that when the similarity between points satisfies a locality property, then the algorithms are plagued by the curse of dimensionality, with respect to the dimensionality of an underlying manifold.

11.1 Introduction

Many semi-supervised learning algorithms rely on the geometry of the data induced by both labeled and unlabeled examples to improve on supervised methods that use only the labeled data. This geometry can be naturally represented by an empirical graph $\mathsf{g} = (V, E)$ where nodes $V = \{1, \dots, n\}$ represent the training data and edges E represent similarities between them (c.f. Section 1.3.3). These similarities are given by a weight matrix $\mathbf{W}$: $\mathbf{W}_{ij}$ is non-zero iff x_i and x_j are "neighbors", i.e. the edge (i, j) is in E (weighted by $\mathbf{W}_{ij}$). The weight matrix $\mathbf{W}$ can be for instance the k-nearest neighbor matrix: $\mathbf{W}_{ij} = 1$ iff x_i is among the k nearest neighbors of

weight matrix

x_j or vice-versa (and is 0 otherwise). Another typical weight matrix is given by the Gaussian kernel of width σ:

$$\mathbf{W}_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}. \tag{11.1}$$

In general, we assume $\mathbf{W}_{ij}$ is given by a symmetric positive function W_X (possibly dependent on the dataset $X = (x_1, \ldots, x_n)$) by $\mathbf{W}_{ij} = W_X(x_i, x_j) \geq 0$. This functional view will be useful in the inductive setting (Section 11.4).

This chapter is organized as follows. In Section 11.2 we present algorithms based on the idea of using the graph structure to spread labels from labeled examples to the whole dataset (Szummer and Jaakkola [2001], Zhu and Ghahramani [2002], Zhou et al. [2004], Zhu et al. [2003]). An alternative approach originating from smoothness considerations yields algorithms based on graph regularization, which naturally leads to a regularization term based on the graph Laplacian (Belkin and Niyogi [2003], Joachims [2003], Zhou et al. [2004], Zhu et al. [2003], Belkin et al. [2004], Delalleau et al. [2005]). This approach, detailed in Section 11.3, is then shown to be tightly linked to the previous label propagation algorithms. In Section 11.4 and Section 11.5 we present two extensions of these algorithms: first a simple way to turn a number of them, originally designed for the transductive setting, into induction algorithms, then a method to better balance classes using prior information about the classes distribution. Section 11.6 finally explores theoretical limitations of these methods which, being based mostly on the local geometry of the data in small neighborhoods, are subject to the curse of dimensionality when the intrinsic dimension of the underlying distribution (the dimensionality of the manifold near which it concentrates) increases, when this manifold is far from being flat.

11.2 Label propagation on a similarity graph

11.2.1 Iterative algorithms

label propagation

Given the graph $\mathtt{g}$, a simple idea for semi-supervised learning is to *propagate labels* on the graph. Starting with nodes $1, 2, \ldots, l$ labeled[1] with their known label (1 or -1) and nodes $l+1, \ldots, n$ labeled with 0, each node starts to propagate its label to its neighbors, and the process is repeated until convergence.

An algorithm of this kind has been proposed by Zhu and Ghahramani [2002], and is described in Algorithm 11.1. Estimated labels on both labeled and unlabeled data are denoted by $\hat{Y} = (\hat{Y}_l, \hat{Y}_u)$, where $\hat{Y}_l$ may be allowed to differ from the given

1. If there are $M > 2$ classes, one can label each node i with a M-dimensional vector (one-hot for labeled samples, i.e. with 0 everywhere except a 1 at index y_i = class of x_i), and use the same algorithms in a one-versus-rest fashion. We consider here the classification case, but extension to regression is straightforward since labels are treated as real values.

Algorithm 11.1 Label propagation (Zhu and Ghahramani [2002])

Compute affinity matrix $\mathbf{W}$ from (11.1)
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Initialize $\hat{Y}^{(0)} \leftarrow (y_1, \ldots, y_l, 0, 0, \ldots, 0)$
Iterate
1. $\hat{Y}^{(t+1)} \leftarrow \mathbf{D}^{-1}\mathbf{W}\hat{Y}^{(t)}$
2. $\hat{Y}_l^{(t+1)} \leftarrow Y_l$

until convergence to $\hat{Y}^{(\infty)}$
Label point x_i by the sign of $\hat{y}_i^{(\infty)}$

labels $Y_l = (y_1, \ldots, y_l)$. In this particular algorithm, $\hat{Y}_l$ is constrained to be equal to Y_l. We propose in Algorithm 11.2 below a slightly different label propagation scheme (originally inspired from the Jacobi iterative method for linear systems), similar to the previous algorithm except that:

- we advocate forcing $\mathbf{W}_{ii} = 0$, which often works better,
- we allow $\hat{Y}_l \neq Y_l$ (which may be useful e.g. when classes overlap), and
- we use an additional regularization term ϵ for better numerical stability.

Algorithm 11.2 Label propagation (inspired from Jacobi iteration algorithm)

Compute an affinity matrix $\mathbf{W}$ such that $\mathbf{W}_{ii} = 0$
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Choose a parameter $\alpha \in (0, 1)$ and a small $\epsilon > 0$
$\mu \leftarrow \frac{\alpha}{1-\alpha} \in (0, +\infty)$
Compute the diagonal matrix $\mathbf{A}$ by $\mathbf{A}_{ii} \leftarrow I_{[l]}(i) + \mu\mathbf{D}_{ii} + \mu\epsilon$
Initialize $\hat{Y}^{(0)} \leftarrow (y_1, \ldots, y_l, 0, 0, \ldots, 0)$
Iterate $\hat{Y}^{(t+1)} \leftarrow \mathbf{A}^{-1}(\mu\mathbf{W}\hat{Y}^{(t)} + \hat{Y}^{(0)})$ until convergence to $\hat{Y}^{(\infty)}$
Label point x_i by the sign of $\hat{y}_i^{(\infty)}$

The iteration step of Algorithm 11.2 can be re-written for a labeled example $(i \leq l)$

$$\hat{y}_i^{(t+1)} \leftarrow \frac{\sum_j \mathbf{W}_{ij}\hat{y}_j^{(t)} + \frac{1}{\mu}y_i}{\sum_j \mathbf{W}_{ij} + \frac{1}{\mu} + \epsilon} \tag{11.2}$$

and for an unlabeled example $(l + 1 \leq i \leq n)$

$$\hat{y}_i^{(t+1)} \leftarrow \frac{\sum_j \mathbf{W}_{ij}\hat{y}_j^{(t)}}{\sum_j \mathbf{W}_{ij} + \epsilon}. \tag{11.3}$$

These two equations can be seen as a weighted average of the neighbors' current labels, where for labeled examples we also add the initial label (whose weight is inversely proportional to the parameter μ). The ϵ parameter is a regularization term to prevent numerical problems when the denominator becomes too small. The convergence of this algorithm follows from the convergence of the Jacobi iteration method for a specific linear system, and will be discussed in Section 11.3.3.

Another similar label propagation algorithm was given by Zhou et al. [2004]: at each step a node i receives a contribution from its neighbors j (weighted by the normalized weight of the edge (i, j)), and an additional small contribution given by its initial value. This process is detailed in Algorithm 11.3 below (the name "label spreading" was inspired from the terminology used by Zhou et al. [2004]). Compared to Algorithm 11.2, it corresponds to the minimization of a slightly different cost criterion, maybe not as intuitive: this will be studied later in Section 11.3.2 and 11.3.3.

Algorithm 11.3 Label spreading (Zhou et al. [2004])

Compute the affinity matrix $\mathbf{W}$ from (11.1) for $i \neq j$ (and $\mathbf{W}_{ii} \leftarrow 0$)
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Compute the normalized graph Laplacian $\mathcal{L} \leftarrow \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
Initialize $\hat{Y}^{(0)} \leftarrow (y_1, \ldots, y_l, 0, 0, \ldots, 0)$
Choose a parameter $\alpha \in [0, 1)$
Iterate $\hat{Y}^{(t+1)} \leftarrow \alpha\mathcal{L}\hat{Y}^{(t)} + (1-\alpha)\hat{Y}^{(0)}$ until convergence to $\hat{Y}^{(\infty)}$
Label point x_i by the sign of $\hat{y}_i^{(\infty)}$

The proof of convergence of Algorithm 11.3 is simple (Zhou et al. [2004]). The iteration equation being $\hat{Y}^{(t+1)} \leftarrow \alpha\mathcal{L}\hat{Y}^{(t)} + (1-\alpha)\hat{Y}^{(0)}$, we have

$$\hat{Y}^{(t+1)} = (\alpha\mathcal{L})^t \hat{Y}^{(0)} + (1-\alpha)\sum_{i=0}^{t}(\alpha\mathcal{L})^i \hat{Y}^{(0)}.$$

The matrix $\mathcal{L}$ being similar to $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W} = \mathbf{D}^{-1/2}\mathcal{L}\mathbf{D}^{1/2}$, it has the same eigenvalues. Since $\mathbf{P}$ is a stochastic matrix by construction, its eigenvalues are in $[-1, 1]$, and consequently the eigenvalues of $\alpha\mathcal{L}$ are in $(-1, 1)$ (remember $\alpha < 1$). It follows that when $t \to \infty$, $(\alpha\mathcal{L})^t \to 0$ and

$$\sum_{i=0}^{t}(\alpha\mathcal{L})^i \to (\mathbf{I} - \alpha\mathcal{L})^{-1}$$

so that

$$\hat{Y}^{(t)} \to \hat{Y}^{(\infty)} = (1-\alpha)(\mathbf{I} - \alpha\mathcal{L})^{-1}\hat{Y}^{(0)}. \tag{11.4}$$

The convergence rate of these three algorithms depends on specific properties of the graph such as the eigenvalues of its Laplacian. In general, we can expect it to be at worst on the order of $O(kn^2)$, where k is the number of neighbors of a point in the graph. In the case of a dense weight matrix, the computational time is thus cubic in n.

11.2.2 Markov random walks

A different algorithm based on label propagation on the similarity graph was proposed earlier by Szummer and Jaakkola [2001]. They consider Markov random walks on the graph with transition probabilities from i to j

transition probabilities

$$p_{ij} = \frac{\mathbf{W}_{ij}}{\sum_k \mathbf{W}_{ik}} \tag{11.5}$$

in order to estimate probabilities of class labels. Here, $\mathbf{W}_{ij}$ is given by a Gaussian kernel for neighbors and 0 for non-neighbors, and $\mathbf{W}_{ii} = 1$ (but one could also use $\mathbf{W}_{ii} = 0$). Each data point x_i is associated with a probability $P(y = 1|i)$ of being of class 1. Given a point x_k, we can compute the probability $P^{(t)}(y_{start} = 1|k)$ that we started from a point of class $y_{start} = 1$ given that we arrived to x_k after t steps of random walk by

$$P^{(t)}(y_{start} = 1|k) = \sum_{i=1}^{n} P(y = 1|i) P_{0|t}(i|k)$$

where $P_{0|t}(i|k)$ is the probability that we started from x_i given that we arrived to k after t steps of random walk (this probability can be computed from the p_{ij}). x_k is then classified to 1 if $P^{(t)}(y_{start} = 1|k) > 0.5$, and to -1 otherwise. The authors propose two methods to estimate the class probabilities $P(y = 1|i)$. One is based on an iterative EM algorithm, the other on maximizing a margin-based criterion, which leads to a closed-form solution (Szummer and Jaakkola [2001]).

It turns out that this algorithm's performance depends crucially on the hyper-parameter t (the length of the random walk). This parameter has to be chosen by cross-validation (if enough data is available) or heuristically (it corresponds intuitively to the amount of propagation we allow in the graph, i.e. to the scale of the clusters we are interested in). An alternative way of using random walks on the graph is to assign to point x_i a label depending on the probability of arriving to a positively labeled example when performing a random walk *starting from* x_i *and until a labeled example is found* (Zhu and Ghahramani [2002], Zhu et al. [2003]). The length of the random walk is not constrained anymore to a fixed value t. In the following, we will show that this probability, denoted by $P(y_{end} = 1|i)$, is equal (up to a shift and scaling) to the label obtained with Algorithm 11.1 (this is similar to the proof by Zhu and Ghahramani [2002]).

When x_i is a labeled example, $P(y_{end} = 1|i) = \delta_{y_i 1}$, and when it is unlabeled we have the relation

$$P(y_{end} = 1|i) = \sum_{j=1}^{n} P(y_{end} = 1|j) p_{ij} \tag{11.6}$$

with the p_{ij} computed as in (11.5). Let us consider the matrix $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, i.e. such that $\mathbf{P}_{ij} = p_{ij}$. We will denote $\hat{z}_i = P(y_{end} = 1|i)$ and $\hat{Z} = (\hat{Z}_l, \hat{Z}_u)$ the corresponding vector split into its labeled and unlabeled parts. Similarly, the matrices $\mathbf{D}$ and $\mathbf{W}$ can be split into four parts:

$$\begin{aligned}\mathbf{D} &= \begin{pmatrix} \mathbf{D}_{ll} & 0 \\ 0 & \mathbf{D}_{uu} \end{pmatrix} \\ \mathbf{W} &= \begin{pmatrix} \mathbf{W}_{ll} & \mathbf{W}_{lu} \\ \mathbf{W}_{ul} & \mathbf{W}_{uu} \end{pmatrix}\end{aligned}$$

Equation (11.6) can then be written

$$\begin{aligned}\hat{Z}_u &= \left(\mathbf{D}_{uu}^{-1}\mathbf{W}_{ul} \mid \mathbf{D}_{uu}^{-1}\mathbf{W}_{uu}\right) \begin{pmatrix} \hat{Z}_l \\ \hat{Z}_u \end{pmatrix} \\ &= \mathbf{D}_{uu}^{-1}\left(\mathbf{W}_{ul}\hat{Z}_l + \mathbf{W}_{uu}\hat{Z}_u\right)\end{aligned}$$

which leads to the linear system

$$L_{uu}\hat{Z}_u = \mathbf{W}_{ul}\hat{Z}_l \tag{11.7}$$

where $L = \mathbf{D} - \mathbf{W}$ is the un-normalized graph Laplacian. Since $\hat{Z}_l$ is known ($\hat{z}_i = 1$ if $y_i = 1$, and 0 otherwise), this linear system can be solved in order to find the probabilities $\hat{Z}_u$ on unlabeled examples. Note that if $(\hat{Z}_u, \hat{Z}_l)$ is a solution of (11.7), then $(\hat{Y}_u, \hat{Y}_l)$ is also a solution, with

$$\begin{aligned}\hat{Y}_u &= 2\hat{Z}_u - (1, 1, \ldots, 1)^\top \\ \hat{Y}_l &= 2\hat{Z}_l - (1, 1, \ldots, 1)^\top = Y_l\end{aligned}$$

This allows us to rewrite the linear system (11.7) in terms of the vector of original labels Y_l as follows:

$$L_{uu}\hat{Y}_u = \mathbf{W}_{ul}\hat{Y}_l \tag{11.8}$$

with the sign of each element y_i of $\hat{Y}_u$ giving the estimated label of x_i (which is equivalent to comparing $\hat{z}_i$ to a 0.5 threshold).

The solution of this random walk algorithm is thus given in closed-form by a linear system, which turns out to be equivalent to iterative Algorithm 11.1 (or equivalently, Algorithm 11.2 when $\mu \to 0$ and $\epsilon = 0$), as we will see in Section 11.3.4.1.

11.3 Quadratic cost criterion

In this section, we investigate semi-supervised learning by minimization of a cost function derived from the graph g. Such methods will be shown to be equivalent to label propagation algorithms presented in the previous section.

11.3.1 Regularization on graphs

The problem of semi-supervised learning on the graph g consists in finding a labeling of the graph that is consistent with both the initial (incomplete) labeling and the geometry of the data induced by the graph structure (edges and weights $\mathbf{W}$). Given a labeling $\hat{Y} = (\hat{Y}_l, \hat{Y}_u)$, consistency with the initial labeling can be measured e.g. by

$$\sum_{i=1}^{l} (\hat{y}_i - y_i)^2 = \|\hat{Y}_l - Y_l\|^2. \tag{11.9}$$

Smoothness Assumption

On the other hand, consistency with the geometry of the data, which follows from the Smoothness (or Manifold) Assumption discussed in Section 1.2, motivates a penalty term of the form

$$\begin{aligned} \frac{1}{2}\sum_{i,j=1}^{n} \mathbf{W}_{ij}(\hat{y}_i - \hat{y}_j)^2 &= \frac{1}{2}\left(2\sum_{i=1}^{n} \hat{y}_i^2 \sum_{j=1}^{n} \mathbf{W}_{ij} - 2\sum_{i,j=1}^{n} \mathbf{W}_{ij}\hat{y}_i\hat{y}_j\right) \\ &= \hat{Y}^\top (\mathbf{D} - \mathbf{W})\hat{Y} \\ &= \hat{Y}^\top L \hat{Y} \end{aligned} \tag{11.10}$$

graph Laplacian

with $L = \mathbf{D} - \mathbf{W}$ the un-normalized graph Laplacian. This means we penalize rapid changes in $\hat{Y}$ between points that are close (as given by the similarity matrix $\mathbf{W}$).

Various algorithms have been proposed based on such considerations. Zhu et al. [2003] force the labels on the labeled data ($\hat{Y}_l = Y_l$) then minimize (11.10) over $\hat{Y}_u$. However, if there is noise in the available labels, it may be beneficial to allow the algorithm to re-label the labeled data (this could also help generalization in a noise-free setting where for instance a positive sample had been drawn from a region of space mainly filled with negative samples). This observation leads to a more general cost criterion involving a trade-off between (11.9) and (11.10) (Belkin et al. [2004], Delalleau et al. [2005]). A small regularization term can also be added in order to prevent degenerate situations, for instance when the graph g has a connected component with no labeled sample. We thus obtain the following general labeling cost[2]:

$$C(\hat{Y}) = \|\hat{Y}_l - Y_l\|^2 + \mu \hat{Y}^\top L \hat{Y} + \mu\epsilon \|\hat{Y}\|^2. \tag{11.11}$$

spectral clustering

Joachims [2003] obtained the same kind of cost criterion from the perspective of spectral clustering. The unsupervised minimization of $\hat{Y}^\top L \hat{Y}$ (under the constraints $\hat{Y}^\top \mathbf{1} = 0$ and $\|\hat{Y}\|^2 = n$) is a relaxation of the NP-hard problem of minimizing the normalized cut of the graph g, i.e. splitting g into two subsets $\mathsf{g}^+ = (V^+, E^+)$ and

2. Belkin et al. [2004] first center the vector Y_l and also constrain $\hat{Y}$ to be centered: these restrictions are needed to obtain theoretical bounds on the generalization error, and will not be discussed in this chapter.

$\mathsf{g}^- = (V^-, E^-)$ such as to minimize

$$\frac{\sum_{i \in V^+, j \in V^-} W_{ij}}{|V^+||V^-|}$$

where the normalization by $|V^+||V^-|$ favors balanced splits. Based on this approach, Joachims [2003] introduced an additional cost which corresponds to our part $\|\hat{Y}_l - Y_l\|^2$ of the cost (11.11), in order to turn this unsupervised minimization into a semi-supervised transductive algorithm (called Spectral Graph Transducer). Note however that although very similar, the solution obtained differs from the straighforward minimization of (11.11) since:

- the labels are not necessarily +1 and -1, but depend on the ratio of the number of positive examples over the number of negative examples (this follows from the normalized cut optimization),
- the constraint $\|\hat{Y}\|^2 = n$ used in the unsupervised setting remains, thus leading to an eigenvalue problem instead of the direct quadratic minimization that will be studied in the next section,
- the eigenspectrum of the graph Laplacian is normalized by replacing the ordered Laplacian eigenvalues by a monotonically increasing function, in order to focus on the ranking among the smallest cuts and abstract, for example, from different magnitudes of edge weights.

graph Laplacian

Belkin and Niyogi [2003] also proposed a semi-supervised algorithm based on the same idea of graph regularization, but using a regularization criterion different from the quadratic penalty term (11.10). It consists in taking advantage of properties of the graph Laplacian L, which can be seen as an operator on functions defined on nodes of the graph g. The graph Laplacian is closely related to the Laplacian on the manifold, whose eigenfunctions provide a basis for the Hilbert space of $\mathcal{L}^2$ functions on the manifold (Rosenberg [1997]). Eigenvalues of the eigenfunctions provide a measure of their smoothness on the manifold (low eigenvalues correspond to smoother functions, with the eigenvalue 0 being associated with the constant function). Projecting any function in $\mathcal{L}^2$ on the first p eigenfunctions (sorted by order of increasing eigenvalue) is thus a way of smoothing it on the manifold. The same principle can be applied to our graph setting, thus leading to Algorithm 11.4 (Belkin and Niyogi [2003]) below. It consists in computing the first p eigenvectors

Algorithm 11.4 Laplacian regularization (Belkin and Niyogi [2003])

Compute affinity matrix $\mathbf{W}$ (with $\mathbf{W}_{ii} = 0$)
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Compute the un-normalized graph Laplacian $L = \mathbf{D} - \mathbf{W}$
Compute the p eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_p$ corresponding to the p smallest eigenvalues of L
Minimize over $a_1, \ldots, a_p$ the quadratic criterion $\sum_{i=1}^{l} \left(y_i - \sum_{j=1}^{p} a_j \mathbf{e}_{j,i} \right)^2$
Label point x_i $(1 \leq i \leq n)$ by the sign of $\sum_{j=1}^{p} a_j \mathbf{e}_{j,i}$

of the graph Laplacian (each eigenvector can be seen as the corresponding eigenfunction applied on training points), then finding the linear combination of these eigenvectors that best predicts the labels (in the mean-squared sense). The idea is to obtain a smooth function (in the sense that it is a linear combination of the p smoothest eigenfunctions of the Laplacian operator on the manifold) that fits the labeled data. This algorithm does not explicitely correspond to the minimization of a non-parametric quadratic criterion such as (11.11) and thus is not covered by the connection shown in Section 11.3.3 with label propagation algorithms, but one must keep in mind that it is based on similar graph regularization considerations and offers competitive classification performance.

11.3.2 Optimization framework

In order to minimize the quadratic criterion (11.11), we can compute its derivative with respect to $\hat{Y}$. We will denote by $\mathbf{S}$ the diagonal matrix $(n \times n)$ given by $\mathbf{S}_{ii} = I_{[l]}(i)$, so that the first part of the cost can be re-written $\|\mathbf{S}\hat{Y} - \mathbf{S}Y\|^2$. The derivative of the criterion is then:

$$\begin{aligned} \frac{1}{2}\frac{\partial C(\hat{Y})}{\partial \hat{Y}} &= \mathbf{S}(\hat{Y} - Y) + \mu L\hat{Y} + \mu\epsilon\hat{Y} \\ &= (\mathbf{S} + \mu L + \mu\epsilon\mathbf{I})\,\hat{Y} - \mathbf{S}Y. \end{aligned}$$

The second derivative is:

$$\frac{1}{2}\frac{\partial^2 C(\hat{Y})}{\partial \hat{Y}\partial \hat{Y}^\top} = \mathbf{S} + \mu L + \mu\epsilon\mathbf{I}$$

which is a positive definite matrix when $\epsilon > 0$ (L is positive semi-definite as shown by (11.10)). This ensures the cost is minimized when the derivative is set to 0, i.e.

$$\hat{Y} = (\mathbf{S} + \mu L + \mu\epsilon\mathbf{I})^{-1}\,\mathbf{S}Y. \tag{11.12}$$

This shows how the new labels can be obtained by a simple matrix inversion. It is interesting to note that this matrix does not depend on the original labels, but only on the graph Laplacian L: the way labels are "propagated" to the rest of the graph is entirely determined by the graph structure.

An alternative (and very similar) criterion was proposed by Zhou et al. [2004], and can be written:

$$\begin{aligned} C'(\hat{Y}) &= \|\hat{Y} - \mathbf{S}Y\|^2 + \frac{\mu}{2}\sum_{i,j}\mathbf{W}_{ij}\left(\frac{\hat{y}_i}{\sqrt{\mathbf{D}_{ii}}} - \frac{\hat{y}_j}{\sqrt{\mathbf{D}_{jj}}}\right)^2 \qquad (11.13)\\ &= \|\hat{Y}_l - Y_l\|^2 + \|\hat{Y}_u\|^2 + \mu\hat{Y}^\top(\mathbf{I} - \mathcal{L})\,\hat{Y} \\ &= \|\hat{Y}_l - Y_l\|^2 + \|\hat{Y}_u\|^2 + \mu\hat{Y}^\top\mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\,\mathbf{D}^{-1/2}\hat{Y} \\ &= \|\hat{Y}_l - Y_l\|^2 + \|\hat{Y}_u\|^2 + \mu(\mathbf{D}^{-1/2}\hat{Y})^\top L(\mathbf{D}^{-1/2}\hat{Y}) \end{aligned}$$

This criterion C' has two main differences with C (11.11):

- the term $\|\hat{Y} - \mathbf{S}Y\|^2 = \|\hat{Y}_l - Y_l\|^2 + \|\hat{Y}_u\|^2$ not only tries to fit the given labels, but also to pull to 0 labels of unlabeled samples (this is a similar but stronger regularization compared to the term $\mu\epsilon\|\hat{Y}\|^2$ in the cost C), and
- labels are normalized by the square root of the degree matrix elements $\mathbf{D}_{ii}$ when computing their similarity. This normalization may not be intuitive, but is necessary for the equivalence with the label propagation Algorithm 11.3, as seen below.

11.3.3 Links with label propagation

The optimization algorithms presented above turn out to be equivalent to the label propagation methods from Section 11.2. Let us first study the optimization of the cost $C(\hat{Y})$ from (11.11). The optimum $\hat{Y}$ is given by (11.12), but another way to obtain this solution, besides matrix inversion, is to solve the linear system using one of the many standard methods available. We focus here on the simple Jacobi iteration method (Saad [1996]), which consists in solving for each component iteratively. Given the system

Jacobi iteration

$$\mathbf{M}x = b \tag{11.14}$$

the approximate solution at step $t+1$ is

$$x_i^{(t+1)} = \frac{1}{\mathbf{M}_{ii}} \left(b - \sum_{j \neq i} \mathbf{M}_{ij} x_j^{(t)} \right). \tag{11.15}$$

Applying this formula with $x := \hat{Y}$, $b := \mathbf{S}Y$ and $\mathbf{M} := \mathbf{S} + \mu L + \mu\epsilon\mathbf{I}$, we obtain:

$$\hat{y}_i^{(t+1)} = \frac{1}{I_{[l]}(i) + \mu \sum_{j \neq i} \mathbf{W}_{ij} + \mu\epsilon} \left(I_{[l]}(i) y_i + \mu \sum_{j \neq i} \mathbf{W}_{ij} \hat{y}_j^{(t)} \right)$$

i.e. exactly the update equations (11.2) and (11.3) used in Algorithm 11.2. Convergence of this iterative algorithm is guaranteed by the following theorem (Saad [1996]): if the matrix $\mathbf{M}$ is strictly diagonally dominant, the Jacobi iteration (11.15) converges to the solution of the linear system (11.14). A matrix $\mathbf{M}$ is strictly diagonally dominant iff $|\mathbf{M}_{ii}| > \sum_{j \neq i} |\mathbf{M}_{ij}|$, which is clearly the case for the matrix $\mathbf{S} + \mu L + \mu\epsilon\mathbf{I}$ (remember $L = \mathbf{D} - \mathbf{W}$ with $\mathbf{D}_{ii} = \sum_{i \neq j} \mathbf{W}_{ij}$, and all $\mathbf{W}_{ij} \geq 0$). Note that this condition also guarantees the convergence of the Gauss-Seidel iteration, which is the same as the Jacobi iteration except that updated coordinates $x_i^{(t+1)}$ are used in the computation of $x_j^{(t+1)}$ for $j > i$. This means we can apply equations (11.2) and (11.3) with $\hat{Y}^{(t+1)}$ and $\hat{Y}^{(t)}$ sharing the same storage.

To show the equivalence between Algorithm 11.3 and the minimization of C' given in (11.13), we compute its derivative with respect to $\hat{Y}$:

$$\frac{1}{2} \frac{\partial C'(\hat{Y})}{\partial \hat{Y}} = \hat{Y} - \mathbf{S}Y + \mu \left(\hat{Y} - \mathcal{L}\hat{Y} \right)$$

and is zero iff

$$\hat{Y} = ((1+\mu)\mathbf{I} - \mu\mathcal{L})^{-1}\,\mathbf{S}Y$$

which is the same equation as (11.4) with $\mu = \alpha/(1-\alpha)$, up to a positive factor (which has no effect on the classification since we use only the sign).

11.3.4 Limit case and analogies

It is interesting to study the limit case when $\mu \to 0$. In this section we will set $\epsilon = 0$ to simplify notations, but one should keep in mind that it is usually better to use a small positive value for regularization. When $\mu \to 0$, the cost (11.11) is dominated by $\|\hat{Y}_l - Y_l\|^2$. Intuitively, this corresponds to

1. forcing $\hat{Y}_l = Y_l$, then
2. minimizing $\hat{Y}^\top L\hat{Y}$.

Writing $\hat{Y} = (Y_l, \hat{Y}_u)$ (i.e. $\hat{Y}_l = Y_l$) and

$$L = \begin{pmatrix} L_{ll} & L_{lu} \\ L_{ul} & L_{uu} \end{pmatrix}$$

the minimization of $\hat{Y}^\top L\hat{Y}$ with respect to $\hat{Y}_u$ leads to

$$L_{ul}Y_l + L_{uu}\hat{Y}_u = 0 \Rightarrow \hat{Y}_u = -L_{uu}^{-1}L_{ul}Y_l. \tag{11.16}$$

If we consider now equation (11.12) where $\hat{Y}_l$ is not constrained anymore, when $\epsilon = 0$ and $\mu \to 0$, using the continuity of the inverse matrix application at $\mathbf{I}$, we obtain that

$$\begin{array}{rcl} \hat{Y}_l & \to & Y_l \\ \hat{Y}_u & = & -L_{uu}^{-1}L_{ul}\hat{Y}_l \end{array}$$

which, as expected, gives us the same solution as (11.16).

11.3.4.1 Analogy with Markov random walks

In Section 11.2.2, we presented an algorithm of label propagation based on Markov random walks on the graph, leading to the linear system (11.8). It is immediate to see that this system is exactly the same as the one obtained in (11.16). The equivalence of the solutions discussed in the previous section between the linear system and iterative algorithms thus shows that the random walk algorithm described in Section 11.2.2 is equivalent to the iterative Algorithm 11.2 when $\mu \to 0$, i.e. when we keep the original labels instead of iteratively updating them by (11.2).

11.3.4.2 Analogy with electric networks

Zhu et al. [2003] also link this solution to heat kernels and give an electric network interpretation taken from Doyle and Snell [1984], which we will present here. This analogy is interesting as it gives a physical interpretation to the optimization and label propagation framework studied in this chapter. Let us consider an electric network built from the graph $\mathtt{g}$ by adding resistors with conductance $\mathbf{W}_{ij}$ between nodes i and j (the conductance is the inverse of the resistance). The positive labeled nodes are connected to a positive voltage source $(+1V)$, the negative ones to a negative voltage source $(-1V)$, and we want to compute the voltage on the *unlabeled* nodes (i.e. their label). Denoting the intensity between i and j by I_{ij}, and the voltage by $V_{ij} = \hat{y}_j - \hat{y}_i$, we will use Ohm's law

$$I_{ij} = \mathbf{W}_{ij} V_{ij} \tag{11.17}$$

and Kirchoff's law on an unlabeled node $i > l$:

$$\sum_j I_{ij} = 0. \tag{11.18}$$

Kirchoff's law states that the sum of currents flowing out from i (such that $I_{ij} > 0$) is equal to the sum of currents flowing into i ($I_{ij} < 0$). Here, it is only useful to apply it to unlabeled nodes as the labeled ones are connected to a voltage source, and thus receive some unknown (and uninteresting) current. Using (11.17), we can rewrite (11.18)

$$\begin{aligned} 0 &= \sum_j \mathbf{W}_{ij}(\hat{y}_j - \hat{y}_i) \\ &= \sum_j \mathbf{W}_{ij}\hat{y}_j - \hat{y}_i \sum_j \mathbf{W}_{ij} \\ &= (\mathbf{W}\hat{Y} - \mathbf{D}\hat{Y})_i \\ &= -(L\hat{Y})_i \end{aligned}$$

and since this is true for all $i > l$, it is equivalent in matrix notations to

$$L_{ul} Y_l + L_{uu} \hat{Y}_u = 0$$

which is exactly (11.16). Thus the solution of the limit case (when labeled examples are forced to keep their given label) is given by the voltage in an electric network where labeled nodes are connected to voltage sources and resistors correspond to weights in the graph $\mathtt{g}$.

11.4 From transduction to induction

inductive setting

The previous algorithms all follow the transduction setting presented in Section 1.2.4. However, it could happen that one needs an inductive algorithm, for instance in a situation where new test examples are presented one at a time and solving the linear system turns out to be too expensive. In such a case, the cost criterion (11.11) naturally leads to an induction formula that can be computed in $O(n)$ time. Assuming that labels $\hat{y}_1, \dots, \hat{y}_n$ have already been computed by one of the algorithms above, and we want the label $\hat{y}$ of a new point x: we can minimize $C(\hat{y}_1, \dots, \hat{y}_n, \hat{y})$ only with respect to this new label $\hat{y}$, i.e. minimize

$$\text{constant} + \mu \left(\sum_j W_X(x, x_j)(\hat{y} - \hat{y}_j)^2 + \epsilon \hat{y}^2 \right)$$

where W_X is the (possibly data-dependent) function that generated the matrix $\mathbf{W}$ on $X = (x_1, \dots, x_n)$. Setting to zero the derivative with respect to $\hat{y}$ directly yields

$$\hat{y} = \frac{\sum_j W_X(x, x_j)\hat{y}_j}{\sum_j W_X(x, x_j) + \epsilon} \tag{11.19}$$

a simple inductive formula whose computational requirements scale linearly with the number of samples already seen.

Parzen windows

It is interesting to note that, if W_X is the k-nearest neighbor function, (11.19) reduces to k-nearest neighbor classification. Similarly, if W_X is the Gaussian kernel (11.1), it is equivalent to the formula for Parzen windows or Nadaraya-Watson non-parametric regression (Nadaraya [1964], Watson [1964]). However, we use in this formula the *learned* predictions on the labeled and unlabeled examples *as if they were observed training values*, instead of relying only on labeled data.

11.5 Incorporating Class Prior Knowledge

From the beginning of the chapter, we have assumed that the class label is given by the sign of $\hat{y}$. Such a rule works well when classes are well separated and balanced. However, if this is not the case (which is likely to happen with real-world datasets), the classification resulting from the label propagations algorithms studied in this chapter may not reflect the prior class distribution.

A way to solve this problem is to perform *class mass normalization* (Zhu et al. [2003]), i.e. to rescale classes so that their respective weights over unlabeled examples match the prior class distribution (estimated from labeled examples). Until now, we had been using a scalar label $\hat{y}_i \in [-1, 1]$, which is handy in the binary case. In this section, for the sake of clarity, we will use a M-dimensional vector (M being the number of classes), with each element $\hat{y}_{i,k}$ between 0 and 1 giving a score (or weight) for class k (see also footnote 1 at the beginning of this chapter). For instance, in the binary case, a scalar $\hat{y}_i \in [-1, 1]$ would be represented by the vector $\left(\frac{1}{2}(1 + \hat{y}_i), \frac{1}{2}(1 - \hat{y}_i)\right)^\top$, where the second element would be the score for class -1.

Class mass normalization works as follows. Let us denote by p_k the prior probability of class k obtained from the *labeled* examples, i.e.

$$p_k = \frac{1}{l}\sum_{i=1}^{l} y_{i,k}.$$

The *mass* of class k as given by our algorithm will be the average of *estimated* weights of class k over unlabeled examples, i.e.

$$m_k = \frac{1}{u}\sum_{i=l+1}^{n} \hat{y}_{i,k}.$$

Class mass normalization consists in scaling each class k by the factor

$$w_k = \frac{p_k}{m_k}$$

i.e. to classify x_i in the class given by $\text{argmax}_k\ w_k\hat{y}_{i,k}$ (instead of the simpler decision function $\text{argmax}_k\ \hat{y}_{i,k}$, equivalent to $\text{sign}(\hat{y}_i)$ in the scalar binary case studied in the previous sections). The goal is to make the scaled masses match the prior class distribution, i.e. after normalization we have that for all k

$$\frac{w_k m_k}{\sum_{j=1}^{M} w_j m_j} = p_k.$$

In general, such a scaling gives a better classification performance *when there are enough labeled data* to accurately estimate the class distribution, and *when the unlabeled data come from the same distribution*. Note also that if there is a m such that each class mass is $m_k = mp_k$, i.e. the masses already reflect the prior class distribution, then the class mass normalization step has no effect, as $w_k = m^{-1}$ for all k.

11.6 Curse of Dimensionality for Semi-Supervised Learning

A large number of the semi-supervised learning algorithms proposed in recent years and discussed in this book are essentially non-parametric local learning algorithms, relying on a neighborhood graph to approximate manifolds near which the data density is assumed to concentrate. It means that the out-of-sample or transductive prediction at x depends mostly on the unlabeled examples very near x and on the labeled examples that are close in the sense of this graph. In this section, we present theoretical arguments that suggest that such methods are unlikely to scale well (in terms of generalization performance) when the intrinsic dimension of these manifolds becomes large (curse of dimensionality), if these manifolds are sufficiently curved (or the functions to learn vary enough).

11.6.1 The Smoothness Prior, Manifold Assumption and Non-Parametric Semi-Supervised Learning

Smoothness and Cluster Assumptions

As introduced in Section 1.2, the *smoothness assumption* (or its semi-supervised variant) about the underlying target function $y(\cdot)$ (such that $y(x_i) = y_i$) is at the core of most of the algorithms studied in this book, along with the *cluster assumption* (or its variant, the *low-density separation assumption*). The former implies that if x_1 is near x_2, then y_1 is expected to be near y_2, and the latter implies that the data density is low near the decision surface. The smoothness assumption is intimately linked to a definition of what it means for x_1 to be near x_2, and that can be embodied in a similarity function on input space, $W_X(\cdot,\cdot)$, which is at the core of the graph-based algorithms reviewed in this chapter, transductive SVMs (where W_X is seen as a kernel), and semi-supervised Gaussian processes (where W_X is seen as the covariance of a prior over functions), both in Part II of this book, as well as the algorithms based on a first unsupervised step to learn a better representation (Part IV).

The central claim of this section is that in order to obtain good results with algorithms that rely solely on the smoothness assumption and on the cluster assumption (or the low-density separation assumption), an acceptable decision surface (in the sense that its error is at an acceptable level) must be "smooth" enough. This can happen if the data for each class lie near a low-dimensional manifold (i.e. the manifold assumption), and these manifolds are smooth enough, i.e., do not have high curvature where it matters, i.e., where a wrong characterization of the manifold would yield to large error rate. This claim is intimately linked to the well known *curse of dimensionality*, so we start the section by reviewing results on generalization error for classical non-parametric learning algorithms as dimension increases. We present theoretical arguments that suggest notions of *locality* of the learning algorithm that make it sensitive to the dimension of the manifold near which data lie. These arguments are not trivial extensions of the arguments for classical non-parametric algorithms, because the semi-supervised algorithms such as those studied in this book involve expansion coefficients (e.g. the $\hat{y}_j$ in equation (11.19)) that are non-local, i.e., the coefficient associated with the j-th example x_j may depend on inputs x_i that are far from x_j, in the sense of the similarity function or kernel $W_X(x_i, x_j)$. For instance, a labeled point x_i far from an unlabeled point x_j (i.e. $W_X(x_i, x_j)$ is small) may still influence the estimated label of x_j if there exists a path in the neighborhood graph $\mathtt{g}$ that connects x_i to x_j (going through unlabeled examples).

non-local learning

In the last sub-section (11.6.5), we will try to argue that it is possible to build *non-local* learning algorithms, while not using very specific priors about the task to be learned. This goes against common folklore that when there are not enough training examples in a given region, one cannot generalize properly in that region. This would suggest that difficult learning problems such as those encountered in Artificial Intelligence (e.g., vision, language, robotics, etc) would benefit from the development of a larger array of such non-local learning algorithms.

In order to discuss the curse of dimensionality for semi-supervised learning, we introduce a particular notion of locality. It applies to learning algorithms that can be labeled as *kernel machines*, i.e., shown to explicitly or implicitly learn a predictor function of the form

kernel machine

$$f(x) = b + \sum_{i=1}^{n} \alpha_i k_X(x, x_i) \tag{11.20}$$

where i runs over all the examples (labeled and unlabeled), and $k_X(\cdot,\cdot)$ is a symmetric function (kernel) that is either chosen a priori or using the whole data set X (and does not need to be positive semi-definite). The learning algorithm is then allowed to choose the scalars b and α_i.

Most of the decision functions learned by the algorithms discussed in this chapter can be written as in (11.20). In particular, the label propagation Algorithm 11.2 leads to the induction formula (11.19) corresponding to

$$\begin{aligned} b &= 0 \\ \alpha_i &= \hat{y}_i \\ k_X(x, x_i) &= \frac{W_X(x, x_i)}{\epsilon + \sum_j W_X(x, x_j)} \end{aligned} \tag{11.21}$$

The Laplacian regularization algorithm (Algorithm 11.4) from Belkin and Niyogi [2003], which first learns about the shape of the manifold with an embedding based on the principal eigenfunctions of the Laplacian of the neighborhood, also falls into this category. As shown by Bengio et al. [2004], the principal eigenfunctions can be estimated by the Nyström formula:

Nyström formula

$$f_k(x) = \frac{\sqrt{n}}{\lambda_k} \sum_{i=1}^{n} v_{k,i} k_X(x, x_i) \tag{11.22}$$

where (λ_k, v_k) is the k-th principal (eigenvalue, eigenvector) pair of the Gram matrix K obtained by $K_{ij} = k_X(x_i, x_j)$, and where $k_X(\cdot,\cdot)$ is a data-dependent equivalent kernel derived from the Laplacian of the neighborhood graph g. Since the resulting decision function is a linear combination of these eigenfunctions, we obtain again a kernel machine (11.20).

In the following, we say that a kernel function $k_X(\cdot,\cdot)$ is **local** if for all $x \in X$, there exists a neighborhood $\mathcal{N}(x) \subset X$ such that

$$f(x) \simeq b + \sum_{x_i \in \mathcal{N}(x)} \alpha_i k_X(x, x_i). \tag{11.23}$$

Intuitively, this means that only the near neighbors of x have a significant contribution to $f(x)$. For instance, if k_X is the Gaussian kernel, $\mathcal{N}(x)$ is defined as the points in X that are close to x with respect to σ (the width of the kernel). If (11.23) is an equality, we say that k_X is **strictly local**. An example is when W_X is the k-nearest neighbor kernel in Algorithm 11.2. k_X obtained by (11.21) is then also the k-nearest neighbor kernel, and we have $\mathcal{N}(x) = \mathcal{N}_k(x)$ the set of the k nearest neighbors of x, so that

$$f(x) = \sum_{x_i \in \mathcal{N}_k(x)} \frac{\hat{y}_i}{k}.$$

Similarly, we say that k_X is **local-derivative** if there exists another kernel $\tilde{k}_X$ such that for all $x \in X$, there exists a neighborhood $\mathcal{N}(x) \subset X$ such that

$$\frac{\partial f}{\partial x}(x) \simeq \sum_{x_i \in \mathcal{N}(x)} \alpha_i (x - x_i) \tilde{k}_X(x, x_i). \tag{11.24}$$

Intuitively, this means that the derivative of f at point x is a vector contained mostly in the span of the vectors $x - x_i$ with x_i a near neighbor of x. For instance, with the Gaussian kernel, we have $k_X(x, x_i) = e^{-\|x - x_i\|^2/2\sigma^2}$ and

$$\frac{\partial k_X(x, x_i)}{\partial x} = -\frac{x - x_i}{\sigma^2} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)$$

so that

$$f(x) \simeq b + \sum_{x_i \in \mathcal{N}(x)} \alpha_i (x - x_i) \left(-\frac{1}{\sigma^2} \exp\left(-\frac{\|x - x_i\|^2}{2\sigma^2}\right)\right).$$

Because here $\tilde{k}_X$ is proportional to a Gaussian kernel with width σ, the neighborhood $\mathcal{N}(x)$ is also defined as the points in X which are close to x with respect to σ. Again, we say that k_X is **strictly local-derivative** when (11.24) is an equality (for instance, when k_X is a thresholded Gaussian kernel, i.e. $k_X(x, x_i) = 0$ when $\|x - x_i\| > \delta$).

11.6.2 Curse of Dimensionality for Classical Non-Parametric Learning

Curse of Dimensionality

The term **curse of dimensionality** has been coined by Bellman [1961] in the context of control problems, but it has been used rightfully to describe the poor generalization performance of local non-parametric estimators as the dimensionality increases. We define *bias* as the square of the expected difference between the estimator and the true target function, and we refer generically to *variance* as the variance of the estimator, in both cases the expectations being taken with respect to the training set as a random variable. It is well known that classical non-parametric estimators must trade bias and variance of the estimator through a smoothness hyper-parameter, e.g. kernel bandwidth σ for the Nadarya-Watson estimator (Gaussian kernel). As σ increases, bias increases and the predictor becomes less local, but variance decreases, hence the *bias-variance dilemma* (Geman et al. [1992]) is also about the *locality* of the estimator.

bias-variance dilemma

A nice property of classical non-parametric estimators is that one can prove their convergence to the target function as $n \to \infty$, i.e. these are consistent estimators. One obtains consistency by appropriately varying the hyper-parameter that controls the locality of the estimator as n increases. Basically, the kernel should be allowed to become more and more local, so that bias goes to zero, but the "effective number of examples" involved in the estimator at x,

$$\frac{1}{\sum_{i=1}^{n} k_X(x, x_i)^2}$$

(equal to k for the k-nearest neighbor estimator, with $k_X(x, x_i) = 1/k$ for x_i a neighbor of x) should increase as n increases, so that variance is also driven to 0. For example one obtains this condition with $\lim_{n\to\infty} k = \infty$ and $\lim_{n\to\infty} \frac{k}{n} = 0$ for the k-nearest neighbor. Clearly the first condition is sufficient for variance to go to 0 and the second for the bias to go to 0 (since k/n is proportional to the volume around x containing the k nearest neighbors). Similarly, for the Nadarya-Watson estimator with bandwidth σ, consistency is obtained if $\lim_{n\to\infty} \sigma = 0$ and $\lim_{n\to\infty} n\sigma = \infty$ (in addition to regularity conditions on the kernel). See the book by Härdle et al. [2004] for a recent and easily accessible exposition (with web version). The bias is due to smoothing the target function over the volume covered by the effective neighbors. As the intrinsic dimensionality of the data increases (the number of dimensions that they actually span locally), bias increases. Since that volume increases exponentially with dimension, the effect of the bias quickly becomes very severe. To see this, consider the classical example of the $[0, 1]^d$ hypercube in $\mathbb{R}^d$ with uniformly distributed data in the hypercube. To hold a fraction p of the data in a sub-cube of it, that sub-cube must have sides of length $p^{1/d}$. As $d \to \infty$, $p^{1/d} \to 1$, i.e. we are averaging over distances that cover almost the whole span of the data, just to keep variance constant (by keeping the effective number of neighbors constant).

For a wide class of kernel estimators with kernel bandwidth σ, the expected generalization error (bias plus variance, ignoring the noise) can be written as follows (Härdle et al. [2004]):

$$\text{expected error} = \frac{C_1}{n\sigma^d} + C_2\sigma^4,$$

with C_1 and C_2 not depending on n nor d. Hence an optimal bandwidth is chosen proportional to $n^{-1/(4+d)}$, and the resulting generalization error converges in $n^{-4/(4+d)}$, which becomes very slow for large d. Consider for example the increase in number of examples required to get the same level of error, in 1 dimension versus d dimensions. If n_1 is the number of examples required to get a level of error e, to get the same level of error in d dimensions requires on the order of $n_1^{(4+d)/5}$ examples, i.e. the **required number of examples is exponential in d**. However, if the data distribution is concentrated on a lower dimensional manifold, it is the **manifold dimension** that matters. Indeed, for data on a smooth lower-dimensional manifold, the only dimension that for instance a k-nearest neighbor classifier sees is the dimension of the manifold, since it only uses the Euclidean distances between the near neighbors, and if they lie on such a manifold then the local Euclidean distances approach the local geodesic distances on the manifold (Tenenbaum et al. [2000]). The curse of dimensionality on a manifold (acting with

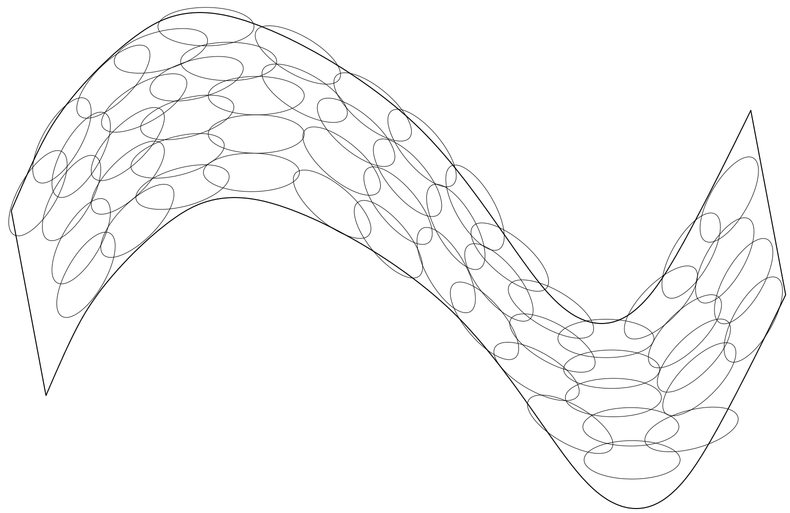

Figure 11.1 Geometric illustration of the effect of the curse of dimensionality on manifolds: the effect depends on the dimension on the manifold, as long as the data are lying strictly on the manifold. In addition to dimensionality, the lack of smoothness (e.g. curvature) of the manifold also has an important influence on the difficulty of generalizing outside of the immediate neighborhood of a training example.

respect to the dimensionality of the manifold) is illustrated in Figure 11.1.

11.6.3 Manifold Geometry: the Curse of Dimensionality for Local Non-Parametric Manifold Learning

Let us first consider how semi-supervised learning algorithms could learn about the shape of the manifolds near which the data concentrate, and how either a high-dimensional manifold or a highly curved manifold could prevent this when the algorithms are local, in the *local-derivative* sense discussed above. As a prototypical example, let us consider the algorithm proposed by Belkin and Niyogi [2003] (Algorithm 11.4). The embedding coordinates are given by the eigenfunctions f_k from (11.22).

The first derivative of f_k with respect to x represents the *tangent vector* of the k-th embedding coordinate. Indeed, it is the direction of variation of x that gives rise locally to the maximal increase in the k-th coordinate. Hence the set of manifold tangent vectors $\{\frac{\partial f_1(x)}{\partial x}, \frac{\partial f_2(x)}{\partial x}, \ldots, \frac{\partial f_d(x)}{\partial x}\}$ spans the estimated **tangent plane** of the manifold.

By the local-derivative property (strict or not), each of the tangent vectors at x is constrained to be exactly or approximately in the span of the difference vectors $x - x_i$, where x_i is a neighbor of x. Hence *the tangent plane is constrained to be a subspace of the span of the vectors* $x - x_i$, with x_i neighbors of x. This is illustrated in Figure 11.2. In addition to the algorithm of Belkin and Niyogi

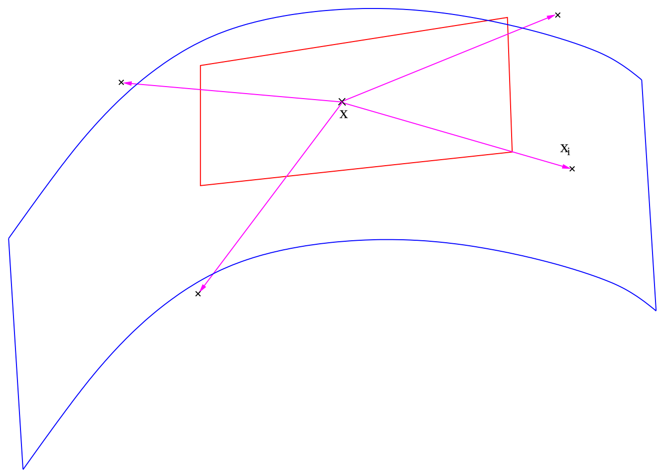

Figure 11.2 Geometric illustration of the effect of the local derivative property shared by semi-supervised graph-based algorithms and spectral manifold learning algorithms. The tangent plane at x is implicitly estimated, and is constrained to be in the span of the vectors $(x_i - x)$, with x_i near neighbors of x. When the number of neighbors is small the estimation of the manifold shape has high variance, but when it is large, the estimation would have high bias unless the true manifold is very flat.

[2003], a number of non-parametric manifold learning algorithms can be shown (e.g. see Bengio et al. [2005]) to have the local derivative property (or the strictly local derivative property): LLE, Isomap, and spectral clustering with Gaussian or nearest-neighbor kernels.

Hence the local-derivative property gives a strong locality constraint to the tangent plane, in particular when the set of neighbors is small. If the number of neighbors is not large in comparison with the manifold dimension, then the locally estimated shape of the manifold will have *high variance*, i.e., we will have a poor estimator of the manifold structure. If the manifold is approximately flat in a large region, then we could simply increase the number of neighbors. However, if the manifold has high curvature, then we cannot increase the number of neighbors without significantly increasing bias in the estimation of the manifold shape. Bias will restrict us to small regions, and the number of such regions could grow exponentially with the dimension of the manifold (Figure 11.1).

A good estimation of the manifold structure – in particular in the region near the decision surface – is crucial for all the graph-based semi-supervised learning algorithms studied in this chapter. It is thanks to a good estimation of the regions in data space where there is high density that we can "propagate labels" in the right places and obtain an improvement with respect to ordinary supervised learning on the labeled examples. The problems due to high curvature and high dimensionality of the manifold are therefore important to consider when applying these graph-based semi-supervised learning algorithms.

11.6.4 Curse of Dimensionality for Local Non-Parametric Semi-Supervised Learning

In this section we focus on algorithms of the type described in Part III of the book (Graph-Based algorithms), using the notation and the induction formula presented in this chapter (on label propagation and a quadratic criterion unifying many of these algorithms).

We consider here that the ultimate objective is to learn a decision surface, i.e. we have a classification problem, and therefore the region of interest in terms of theoretical analysis is mostly the region near the decision surface. For example, if we do not characterize the manifold structure of the underlying distribution in a region far from the decision surface, it is not important, as long as we get it right near the decision surface. Whereas in the previous section we built an argument based on capturing the shape of the manifold associated with each class, here we focus directly on the discriminant function and on learning the shape of the decision surface.

An intuitive view of label propagation suggests that a region of the manifold around a labeled (e.g. positive) example will be entirely labeled positively, as the example spreads its influence by propagation on the graph representing the underlying manifold. Thus, the number of regions with constant label should be on the same order as (or less than) the number of labeled examples. This is easy to see in the case of a sparse weight matrix $\mathbf{W}$, i.e. when the affinity function is *strictly local*. We define a region with constant label as a connected subset of the graph g where all nodes x_i have the same estimated label (sign of $\hat{y}_i$), and such that no other node can be added while keeping these properties. The following proposition then holds (note that it is also true, but trivial, when $\mathbf{W}$ defines a fully connected graph, i.e. $\mathcal{N}(x) = X$ for all x).

Proposition 11.1 *After running a label propagation algorithm minimizing a cost of the form* (11.11)*, the number of regions with constant estimated label is less than (or equal to) the number of labeled examples.*

Proof By contradiction, if this proposition is false, then there exists a region with constant estimated label that does not contain any labeled example. Without loss of generality, consider the case of a positive constant label, with $x_{l+1}, \ldots, x_{l+q}$ the q samples in this region. The part of the cost (11.11) depending on their labels is

$$\begin{aligned} C(\hat{y}_{l+1}, \ldots, \hat{y}_{l+q}) = & \quad \frac{\mu}{2} \sum_{i,j=l+1}^{l+q} \mathbf{W}_{ij} (\hat{y}_i - \hat{y}_j)^2 \\ & + \quad \mu \sum_{i=l+1}^{l+q} \left(\sum_{j \notin \{l+1,\ldots,l+q\}} \mathbf{W}_{ij} (\hat{y}_i - \hat{y}_j)^2 \right) \\ & + \quad \mu\epsilon \sum_{i=l+1}^{l+q} \hat{y}_i^2. \end{aligned}$$

The second term is stricly positive, and because the region we consider is maximal (by definition) all samples x_j outside of the region such that $\mathbf{W}_{ij} > 0$ verify $\hat{y}_j < 0$ (for x_i a sample in the region). Since all $\hat{y}_i$ are stricly positive for $i \in \{l+1, \ldots, l+q\}$, this means this second term can be stricly decreased by setting all $\hat{y}_i$ to 0 for $i \in \{l+1, \ldots, l+q\}$. This also sets the first and third terms to zero (i.e. their minimum), showing that the set of labels $\hat{y}_i$ are not optimal, which is in contradiction with their definition as the labels that minimize C. ■

This means that if the class distributions are such that there are many distinct regions with constant labels (either separated by low-density regions or regions with samples from the other class), we will need at least the same number of labeled samples as there are such regions (assuming we are using a strictly local kernel such as the k-nearest neighbor kernel, or a thresholded Gaussian kernel). But this number could *grow exponentially with the dimension of the manifold(s) on which the data lie*, for instance in the case of a labeling function varying highly along each dimension, *even if the label variations are "simple" in a non-local sense*, e.g. if they alternate in a regular fashion.

When the affinity matrix $\mathbf{W}$ is not sparse (e.g. Gaussian kernel), obtaining such a result is less obvious. However, for local kernels, there often exists a sparse approximation of $\mathbf{W}$ (for instance, in the case of a Gaussian kernel, one can set to 0 entries below a given threshold or that do not correspond to a k-nearest neighbor relationship). Thus we conjecture the same kind of result holds for such dense weight matrices obtained from a local kernel.

Another indication that highly varying functions are fundamentally hard to learn with graph-based semi-supervised learning algorithms is given by the following theorem (Bengio et al. [2006a]):

Theorem 11.2 *Suppose that the learning problem is such that in order to achieve a given error level for samples from a distribution P with a Gaussian kernel machine (11.20), then f must change sign at least $2k$ times along some straight line (i.e., in the case of a classifier, the decision surface must be crossed at least $2k$ times by that straight line). Then the kernel machine must have at least k examples (labeled or unlabeled).*

The theorem is proven for the case where k_X is the Gaussian kernel, but we conjecture that the same result applies to other local kernels, such as the normalized Gaussian or the k-nearest-neighbor kernels implicitly used in graph-based semi-supervised learning algorithms. It is coherent with Proposition 11.1 since both tell us that we need at least k examples to represent k "variations" in the underlying target classifier, whether along a straight line or as the number of regions of differing class on a manifold.

11.6.5 Outlook: Non-Local Semi-Supervised Learning

What conclusions should we draw from the previous results? They should help to better circumscribe where the current local semi-supervised learning algorithms are likely to be most effective, and they should also help to suggest directions of research into non-local learning algorithms, either using non-local kernels or similarity functions, or using altogether other principles of generalization.

When applying a local semi-supervised learning algorithm to a new task, one should consider the plausibility of the hypothesis of a low-dimensional manifold near which the distribution concentrates. For some problems this could be very reasonable a priori (e.g. printed digit images vary mostly due to a few geometric and optical effects). For others, however, one would expect tens or hundreds of degrees of freedom (e.g., many Artificial Intelligence problems, such as natural language processing or recognition of complex composite objects).

Concerning new directions of research suggested by these results, several possible approaches can already be mentioned:

- Semi-supervised algorithms that are not based on the neighborhood graph, such as the one presented in Chapter 9, in which a discriminant training criterion for supervised learning is adapted to semi-supervised learning by taking advantage of the *cluster hypothesis*, more precisely, the *low-density separation hypothesis* (see Section 1.2),
- Algorithms based on the neighborhood graph but in which the kernel or similarity function (a) is non-isotropic (b) is adapted based on the data (with the spread in different directions being adapted). In that case the predictor will not be either local nor local-derivative. More generally, the structure of the similarity function at x should be inferred based not just on the training data in the close neighborhood of x. For an example of such non-local learning in the unsupervised setting, see Bengio and Monperrus [2005], Bengio et al. [2006b].
- Other data-dependent kernels could be investigated, but one should check whether the adaptation allows non-local learning, i.e. that information at x could be used to usefully alter the prediction at a point x' far from x.
- More generally, algorithms that *learn a similarity function* $Sim(x, y)$ in a non-local way (i.e. taking advantage of examples far from x and y) should be good candidates to consider to defeat the curse of dimensionality.

11.7 Discussion

This chapter shows how different graph-based semi-supervised learning algorithms can be cast into a common framework of label propagation and quadratic criterion optimization. They benefit from both points of view: the iterative label propagation methods can provide simple efficient approximate solutions, while the analysis of the quadratic criterion helps to understand what these algorithms really do. The solution can also be linked to physical phenomena such as voltage in an electric network built from the graph, which provides other ways to reason about this problem. In addition, the optimization framework leads to a natural extension of the inductive setting that is closely related to other classical non-parametric learning algorithms such as k-nearest neighbor or Parzen windows. Induction will be studied in more depth in the next chapter, and the induction formula (11.19) will turn out to be the basis for a subset approximation algorithm presented in Chapter 18. Finally, we have shown that the local semi-supervised learning algorithms are likely to be limited to learning smooth functions for data living near low dimensional manifolds. Our approach of locality properties suggests a way to check whether new semi-supervised learning algorithms have a chance to scale to higher dimensional tasks or learning less smooth functions, and motivates further investigation in non-local learning algorithms.

Acknowledgments

The authors would like to thank the editors and anonymous reviewers for their helpful comments and suggestions. This chapter has also greatly benefited from advice from Mikhail Belkin, Dengyong Zhou and Xiaojin Zhu, whose papers first motivated this research (Belkin and Niyogi [2003], Zhou et al. [2004], Zhu et al. [2003]). The authors also thank the following funding organizations for their financial support: Canada Research Chair, NSERC and MITACS.

References

Y. S. Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272(4):64–69, 1995.

A. K. Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16:373–379, 1970.

S. Amari and S. Wu. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks*, 12(6):783–789, 1999.

M. Belkin, I. Matveeva, and P. Niyogi. Regularization and semi-supervised learning on large graphs. In *COLT*, 2004.

M. Belkin and P. Niyogi. Using manifold structure for partially labeled classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, New Jersey, 1961.

Y. Bengio, O. Delalleau, and N. Le Roux. The curse of dimensionality for local kernel machines. Technical Report 1258, Département d'informatique et recherche opérationnelle, Université de Montréal, 2005.

Y. Bengio, O. Delalleau, and N. Le Roux. The curse of highly variable functions for local kernel machines. In *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006a.

Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.

Y. Bengio, H. Larochelle, and P. Vincent. Non-local manifold parzen windows. In *Advances in Neural Information Processing Systems 18*. MIT Press, Cambridge, MA, 2006b.

Y. Bengio and M. Monperrus. Non-local manifold tangent learning. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*. MIT Press, 2005.

O. Bousquet, O. Chapelle, and M. Hein. Measure based regularization. In *NIPS*, Cambridge, MA, USA, 2004. MIT Press.

O. Chapelle and A. Zien. Semi-supervised classification by low density separation. In *Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

T. F. Cox and M. A. Cox. *Multidimensional Scaling*. Chapman & Hall, 1994.

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

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

B. Fischer, V. Roth, and J. M. Buhmann. Clustering with the connectivity kernel. In *NIPS*, volume 16, 2004.

S. C. Fralick. Learning to recognize patterns wothout a teacher. *IEEE Transactions on Information Theory*, 13:57–64, 1967.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

B. Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE TPAMI*, 2004. In press.

W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*.

Springer, http://www.xplore-stat.de/ebooks/ebooks.html, 2004.

T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.

G. Lebanon. Learning riemannian metrics. In *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

E. A. Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9:141–142, 1964.

C. S. Ong, X. Mary, S. Canu, and A. J. Smola. Learning with non-positive kernels. In *ICML*, pages 639–646, 2004.

M. Ouimet and Y. Bengio. Greedy spectral embedding. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

S. Rosenberg. *The Laplacian on a Riemannian Manifold*. Cambridge University Press, Cambridge, UK, 1997.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, MA, 1996.

L. K. Saul and M. I. Jordan. A variational model for model-based interpolation. In *NIPS*, volume 9, 1997.

H. J. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11:363–371, 1965.

M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. In *NIPS*, volume 14, 2001.

J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

P. Vincent and Y. Bengio. Density-sensitive metrics and kernels. Presented at the Snowbird Learning Workshop, 2003.

G. S. Watson. Smooth regression analysis. *Sankhya - The Indian Journal of Statistics*, 26:359–372, 1964.

C. K. I. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In T.K. Leen, T.G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 682–688, Cambridge, MA, 2001. MIT Press.

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, volume 16, 2004.

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

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