Introduction to Semi-Supervised Learning

Synthesis Lectures on Artificial Intelligence and Machine Learning

Introduction to Semi-Supervised Learning

Xiaojin Zhu and Andrew B. Goldberg

www.morganclaypool.com

ISBN: 9781598295474 paperback
ISBN: 9781598295481 ebook

DOI 10.2200/S00196ED1V01Y200906AIM006

A Publication in the Morgan & Claypool Publishers series
SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Lecture #6
Series Editors: Ronald J. Brachman, *Yahoo! Research*
Thomas Dietterich, *Oregon State University*

Series ISSN
Synthesis Lectures on Artificial Intelligence and Machine Learning
Print 1939-4608 Electronic 1939-4616

Introduction to Semi-Supervised Learning

Xiaojin Zhu and Andrew B. Goldberg
University of Wisconsin, Madison

SYNTHESIS LECTURES ON ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING #6

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Semi-supervised learning is a learning paradigm concerned with the study of how computers and natural systems such as humans learn in the presence of both labeled and unlabeled data. Traditionally, learning has been studied either in the unsupervised paradigm (e.g., clustering, outlier detection) where all the data is unlabeled, or in the supervised paradigm (e.g., classification, regression) where all the data is labeled. The goal of semi-supervised learning is to understand how combining labeled and unlabeled data may change the learning behavior, and design algorithms that take advantage of such a combination. Semi-supervised learning is of great interest in machine learning and data mining because it can use readily available unlabeled data to improve supervised learning tasks when the labeled data is scarce or expensive. Semi-supervised learning also shows potential as a quantitative tool to understand human category learning, where most of the input is self-evidently unlabeled. In this introductory book, we present some popular semi-supervised learning models, including self-training, mixture models, co-training and multiview learning, graph-based methods, and semi-supervised support vector machines. For each model, we discuss its basic mathematical formulation. The success of semi-supervised learning depends critically on some underlying assumptions. We emphasize the assumptions made by each model and give counterexamples when appropriate to demonstrate the limitations of the different models. In addition, we discuss semi-supervised learning for cognitive psychology. Finally, we give a computational learning theoretic perspective on semi-supervised learning, and we conclude the book with a brief discussion of open questions in the field.

KEYWORDS

semi-supervised learning, transductive learning, self-training, Gaussian mixture model, expectation maximization (EM), cluster-then-label, co-training, multiview learning, mincut, harmonic function, label propagation, manifold regularization, semi-supervised support vector machines (S3VM), transductive support vector machines (TSVM), entropy regularization, human semi-supervised learning

To our parents

Yu and Jingquan
Susan and Steven Goldberg

with much love and gratitude.

Contents

Preface

The book is a beginner's guide to semi-supervised learning. It is aimed at advanced undergraduates, entry-level graduate students and researchers in areas as diverse as Computer Science, Electrical Engineering, Statistics, and Psychology. The book assumes that the reader is familiar with elementary calculus, probability and linear algebra. It is helpful, but not necessary, for the reader to be familiar with statistical machine learning, as we will explain the essential concepts in order for this book to be self-contained. Sections containing more advanced materials are marked with a star. We also provide a basic mathematical reference in Appendix A.

Our focus is on semi-supervised model assumptions and computational techniques. We intentionally avoid competition-style benchmark evaluations. This is because, in general, semi-supervised learning models are sensitive to various settings, and no benchmark that we know of can characterize the full potential of a given model on all tasks. Instead, we will often use simple artificial problems to "break" the models in order to reveal their assumptions. Such analysis is not frequently encountered in the literature.

Semi-supervised learning has grown into a large research area within machine learning. For example, a search for the phrase "semi-supervised" in May 2009 yielded more than 8000 papers in Google Scholar. While we attempt to provide a basic coverage of semi-supervised learning, the selected topics are not able to reflect the most recent advances in the field. We provide a "bibliographical notes" section at the end of each chapter for the reader to dive deeper into the topics.

We would like to express our sincere thanks to Thorsten Joachims and the other reviewers for their constructive reviews that greatly improved the book. We thank Robert Nowak for his excellent learning theory lecture notes, from which we take some materials for Section 8.1. Our thanks also go to Bryan Gibson, Tushar Khot, Robert Nosofsky, Timothy Rogers, and Zhiting Xu for their valuable comments.

We hope you enjoy the book.

Xiaojin Zhu and Andrew B. Goldberg
Madison, Wisconsin

CHAPTER 1

Introduction to Statistical Machine Learning

We start with a gentle introduction to statistical machine learning. Readers familiar with machine learning may wish to skip directly to Section 2, where we introduce semi-supervised learning.

Example 1.1. You arrive at an extrasolar planet and are welcomed by its resident little green men. You observe the weight and height of 100 little green men around you, and plot the measurements in Figure 1.1. What can you learn from this data?

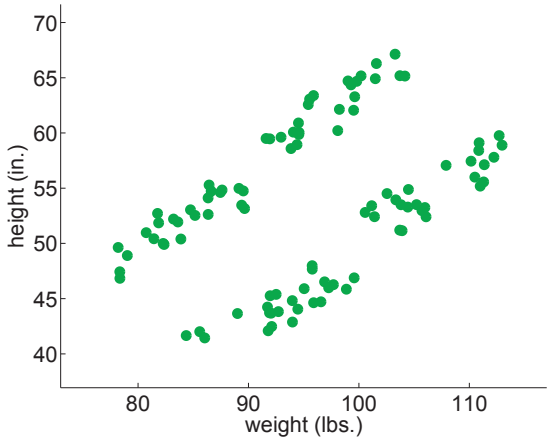


Figure 1.1: The weight and height of 100 little green men from the extrasolar planet. Each green dot is an instance, represented by two features: weight and height.

This is a typical example of a machine learning scenario (except the little green men part). We can perform several tasks using this data: group the little green men into subcommunities based on weight and/or height, identify individuals with extreme (possibly erroneous) weight or height values, try to predict one measurement based on the other, etc. Before exploring such machine learning tasks, let us begin with some definitions.

1.1 THE DATA

Definition 1.2. Instance. An *instance* $\mathbf{x}$ represents a specific object. The instance is often represented by a D-dimensional *feature vector* $\mathbf{x} = (x_1, \ldots, x_D) \in \mathbb{R}^D$, where each dimension is called a *feature*. The length D of the feature vector is known as the dimensionality of the feature vector.

The feature representation is an abstraction of the objects. It essentially ignores all other information not represented by the features. For example, two little green men with the same weight and height, but with different names, will be regarded as indistinguishable by our feature representation. Note we use boldface $\mathbf{x}$ to denote the whole instance, and x_d to denote the d-th feature of $\mathbf{x}$. In our example, an instance is a specific little green man; the feature vector consists of $D = 2$ features: x_1 is the weight, and x_2 is the height. Features can also take discrete values. When there are multiple instances, we will use x_{id} to denote the i-th instance's d-th feature.

Definition 1.3. Training Sample. A *training sample* is a collection of instances $\{\mathbf{x}_i\}_{i=1}^n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, which acts as the input to the learning process. We assume these instances are sampled independently from an underlying distribution $P(\mathbf{x})$, which is unknown to us. We denote this by $\{\mathbf{x}_i\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} P(\mathbf{x})$, where i.i.d. stands for independent and identically distributed.

In our example, the training sample consists of $n = 100$ instances $\mathbf{x}_1, \ldots, \mathbf{x}_{100}$. A training sample is the "experience" given to a learning algorithm. What the algorithm can learn from it, however, varies. In this chapter, we introduce two basic learning paradigms: *unsupervised learning* and *supervised learning*.

1.2 UNSUPERVISED LEARNING

Definition 1.4. Unsupervised learning. Unsupervised learning algorithms work on a training sample with n instances $\{\mathbf{x}_i\}_{i=1}^n$. There is no teacher providing supervision as to how individual instances should be handled—this is the defining property of unsupervised learning. Common unsupervised learning tasks include:

- clustering, where the goal is to separate the n instances into groups;
- novelty detection, which identifies the few instances that are very different from the majority;
- dimensionality reduction, which aims to represent each instance with a lower dimensional feature vector while maintaining key characteristics of the training sample.

Among the unsupervised learning tasks, the one most relevant to this book is *clustering*, which we discuss in more detail.

Definition 1.5. Clustering. Clustering splits $\{\mathbf{x}_i\}_{i=1}^n$ into k clusters, such that instances in the same cluster are similar, and instances in different clusters are dissimilar. The number of clusters k may be specified by the user, or may be inferred from the training sample itself.

How many clusters do you find in the little green men data in Figure 1.1? Perhaps $k = 2$, $k = 4$, or more. Without further assumptions, either one is acceptable. Unlike in supervised learning (introduced in the next section), there is no teacher that tells us which instances should be in each cluster.

There are many clustering algorithms. We introduce a particularly simple one, *hierarchical agglomerative clustering*, to make unsupervised learning concrete.

Algorithm 1.6. Hierarchical Agglomerative Clustering.

Input*: a training sample* $\{\mathbf{x}_i\}_{i=1}^n$*; a distance function* $d()$.
1. Initially, place each instance in its own cluster (called a singleton cluster).
2. ***while*** *(number of clusters* > 1*)* ***do****:*
3. Find the closest cluster pair A, B*, i.e., they minimize* $d(A, B)$.
4. Merge A, B *to form a new cluster.*
Output*: a binary tree showing how clusters are gradually merged from singletons to a root cluster, which contains the whole training sample.*

This clustering algorithm is simple. The only thing unspecified is the distance function $d()$. If $\mathbf{x}_i, \mathbf{x}_j$ are two singleton clusters, one way to define $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between them:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{s=1}^{D} (x_{is} - x_{js})^2}. \tag{1.1}$$

We also need to define the distance between two non-singleton clusters A, B. There are multiple possibilities: one can define it to be the distance between the closest pair of points in A and B, the distance between the farthest pair, or some average distance. For simplicity, we will use the first option, also known as *single linkage*:

$$d(A, B) = \min_{\mathbf{x} \in A, \mathbf{x}' \in B} d(\mathbf{x}, \mathbf{x}'). \tag{1.2}$$

It is not necessary to fully grow the tree until only one cluster remains: the clustering algorithm can be stopped at any point if $d()$ exceeds some threshold, or the number of clusters reaches a predetermined number k.

Figure 1.2 illustrates the results of hierarchical agglomerative clustering for $k = 2, 3, 4$, respectively. The clusters certainly look fine. But because there is no information on how each instance *should* be clustered, it can be difficult to objectively evaluate the result of clustering algorithms.

1.3 SUPERVISED LEARNING

Suppose you realize that your alien hosts have a gender: female or male (so they should not all be called little green men after all). You may now be interested in predicting the gender of a particular

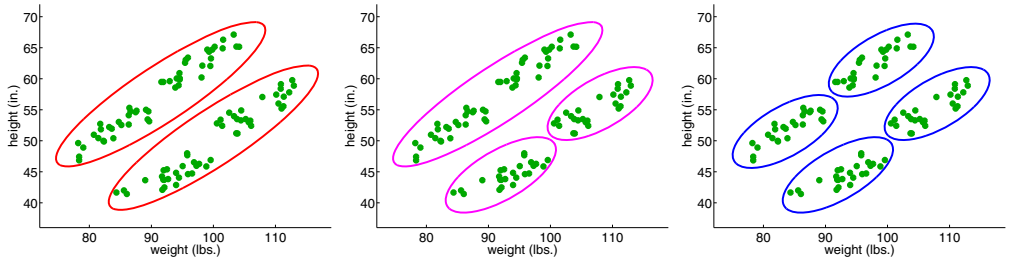


Figure 1.2: Hierarchical agglomerative clustering results for $k = 2, 3, 4$ on the 100 little green men data.

alien from his or her weight and height. Alternatively, you may want to predict whether an alien is a juvenile or an adult using weight and height. To explain how to approach these tasks, we need more definitions.

Definition 1.7. Label. A label y is the desired prediction on an instance $\mathbf{x}$.

Labels may come from a finite set of values, e.g., {female, male}. These distinct values are called *classes*. The classes are usually encoded by integer numbers, e.g., female $= -1$, male $= 1$, and thus $y \in \{-1, 1\}$. This particular encoding is often used for binary (two-class) labels, and the two classes are generically called the negative class and the positive class, respectively. For problems with more than two classes, a traditional encoding is $y \in \{1, \ldots, C\}$, where C is the number of classes. In general, such encoding does not imply structure in the classes. That is to say, the two classes encoded by $y = 1$ and $y = 2$ are not necessarily closer than the two classes $y = 1$ and $y = 3$. Labels may also take continuous values in $\mathbb{R}$. For example, one may attempt to predict the blood pressure of little green aliens based on their height and weight.

In supervised learning, the training sample consists of pairs, each containing an instance $\mathbf{x}$ and a label y: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. One can think of y as the label on $\mathbf{x}$ provided by a teacher, hence the name *supervised* learning. Such (instance, label) pairs are called *labeled data*, while instances alone without labels (as in unsupervised learning) are called *unlabeled data*. We are now ready to define supervised learning.

Definition 1.8. Supervised learning. Let the domain of instances be $\mathcal{X}$, and the domain of labels be $\mathcal{Y}$. Let $P(\mathbf{x}, y)$ be an (unknown) joint probability distribution on instances and labels $\mathcal{X} \times \mathcal{Y}$. Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \overset{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$, supervised learning trains a function $f : \mathcal{X} \mapsto \mathcal{Y}$ in some function family $\mathcal{F}$, with the goal that $f(\mathbf{x})$ predicts the true label y on future data $\mathbf{x}$, where $(\mathbf{x}, y) \overset{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$ as well.

Depending on the domain of label y, supervised learning problems are further divided into *classification* and *regression*:

Definition 1.9. Classification. Classification is the supervised learning problem with discrete classes $\mathcal{Y}$. The function f is called a *classifier*.

Definition 1.10. Regression. Regression is the supervised learning problem with continuous $\mathcal{Y}$. The function f is called a *regression function*.

What exactly is a good f? The best f is by definition

$$f^* = \underset{f \in \mathcal{F}}{\operatorname{argmin}} \, \mathbb{E}_{(\mathbf{x}, y) \sim P} \left[c(\mathbf{x}, y, f(\mathbf{x})) \right], \tag{1.3}$$

where argmin means "finding the f that minimizes the following quantity". $\mathbb{E}_{(\mathbf{x}, y) \sim P} [\cdot]$ is the expectation over random test data drawn from P. Readers not familiar with this notation may wish to consult Appendix A. $c(\cdot)$ is a *loss function* that determines the cost or impact of making a prediction $f(\mathbf{x})$ that is different from the true label y. Some typical loss functions will be discussed shortly. Note we limit our attention to some function family $\mathcal{F}$, mostly for computational reasons. If we remove this limitation and consider all possible functions, the resulting f^* is the *Bayes optimal predictor*, the best one can hope for on average. For the distribution P, this function will incur the lowest possible loss when making predictions. The quantity $\mathbb{E}_{(\mathbf{x}, y) \sim P}[c(\mathbf{x}, y, f^*(\mathbf{x}))]$ is known as the *Bayes error*. However, the Bayes optimal predictor may not be in $\mathcal{F}$ in general. Our goal is to find the $f \in \mathcal{F}$ that is as close to the Bayes optimal predictor as possible.

It is worth noting that the underlying distribution $P(\mathbf{x}, y)$ is unknown to us. Therefore, it is not possible to directly find f^*, or even to measure any predictor f's performance, for that matter. Here lies the fundamental difficulty of statistical machine learning: one has to *generalize* the prediction from a finite training sample to any unseen test data. This is known as *induction*.

To proceed, a seemingly reasonable approximation is to gauge f's performance using training sample error. That is, to replace the unknown expectation by the average over the training sample:

Definition 1.11. Training sample error. Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, the training sample error is

$$\frac{1}{n} \sum_{i=1}^{n} c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)). \tag{1.4}$$

For classification, one commonly used loss function is the 0-1 loss $c(\mathbf{x}, y, f(\mathbf{x})) \equiv (f(\mathbf{x}_i) \neq y_i)$:

$$\frac{1}{n} \sum_{i=1}^{n} (f(\mathbf{x}_i) \neq y_i), \tag{1.5}$$

where $f(\mathbf{x}) \neq y$ is 1 if f predicts a different class than y on x, and 0 otherwise. For regression, one commonly used loss function is the squared loss $c(\mathbf{x}, y, f(\mathbf{x})) \equiv (f(\mathbf{x}_i) - y_i)^2$:

$$\frac{1}{n}\sum_{i=1}^{n}(f(\mathbf{x}_i) - y_i)^2. \tag{1.6}$$

Other loss functions will be discussed as we encounter them later in the book.

It might be tempting to seek the f that minimizes training sample error. However, this strategy is flawed: such an f will tend to *overfit* the particular training sample. That is, it will likely fit itself to the statistical noise in the particular training sample. It will learn more than just the true relationship between $\mathcal{X}$ and $\mathcal{Y}$. Such an overfitted predictor will have small training sample error, but is likely to perform less well on future test data. A sub-area within machine learning called computational learning theory studies the issue of overfitting. It establishes rigorous connections between the training sample error and the true error, using a formal notion of complexity such as the Vapnik-Chervonenkis dimension or Rademacher complexity. We provide a concise discussion in Section 8.1. Informed by computational learning theory, one reasonable training strategy is to seek an f that "almost" minimizes the training sample error, while *regularizing* f so that it is not too complex in a certain sense. Interested readers can find the references in the bibliographical notes.

To estimate f's future performance, one can use a separate sample of labeled instances, called the *test sample*: $\{(\mathbf{x}_j, y_j)\}_{j=n+1}^{n+m} \overset{\text{i.i.d.}}{\sim} P(\mathbf{x}, y)$. A test sample is not used during training, and therefore provides a faithful (unbiased) estimation of future performance.

Definition 1.12. Test sample error. The corresponding test sample error for classification with 0-1 loss is

$$\frac{1}{m}\sum_{j=n+1}^{n+m}(f(\mathbf{x}_j) \neq y_j), \tag{1.7}$$

and for regression with squared loss is

$$\frac{1}{m}\sum_{j=n+1}^{n+m}(f(\mathbf{x}_j) - y_j)^2. \tag{1.8}$$

In the remainder of the book, we focus on classification due to its prevalence in semi-supervised learning research. Most ideas discussed also apply to regression, though.

As a concrete example of a supervised learning method, we now introduce a simple classification algorithm: k-nearest-neighbor (kNN).

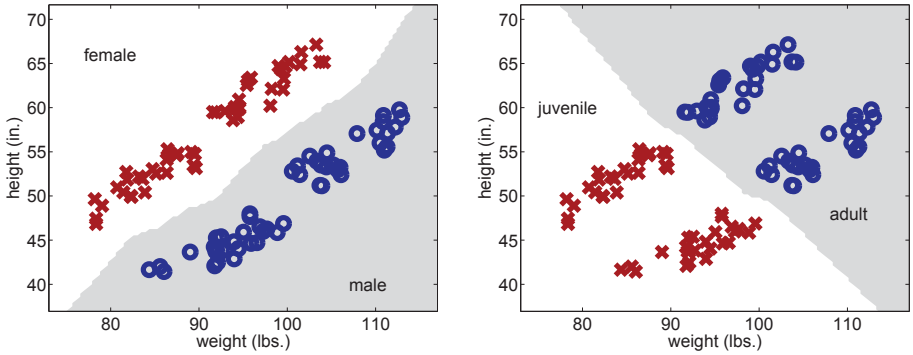
Algorithm 1.13. k-nearest-neighbor classifier.

Input: Training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$*; distance function* $d()$*;*
number of neighbors k*; test instance* $\mathbf{x}^*$

1. Find the k training instances $\mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}$ closest to $\mathbf{x}^*$ under distance $d()$.
2. Output y^* as the majority class of $y_{i_1}, \ldots, y_{i_k}$. Break ties randomly.

Being a D-dimensional feature vector, the test instance $\mathbf{x}^*$ can be viewed as a point in D-dimensional feature space. A classifier assigns a label to each point in the feature space. This divides the feature space into decision regions within which points have the same label. The boundary separating these regions is called the *decision boundary* induced by the classifier.

Example 1.14. Consider two classification tasks involving the little green aliens. In the first task in Figure 1.3(a), the task is gender classification from weight and height. The symbols are training data. Each training instance has a label: female (red cross) or male (blue circle). The decision regions from a 1NN classifier are shown as white and gray. In the second task in Figure 1.3(b), the task is age classification on the same sample of training instances. The training instances now have different labels: juvenile (red cross) or adult (blue circle). Again, the decision regions of 1NN are shown. Notice that, for the same training instances but different classification goals, the decision boundary can be quite different. Naturally, this is a property unique to supervised learning, since unsupervised learning does not use any particular set of labels at all.



(a) classification by gender (b) classification by age

Figure 1.3: Classify by gender or age from a training sample of 100 little green aliens, with 1-nearest-neighbor decision regions shown.

In this chapter, we introduced statistical machine learning as a foundation for the rest of the book. We presented the unsupervised and supervised learning settings, along with concrete examples of each. In the next chapter, we provide an overview of semi-supervised learning, which falls somewhere between these two. Each subsequent chapter will present specific families of semi-supervised learning algorithms.

BIBLIOGRAPHICAL NOTES

There are many excellent books written on statistical machine learning. For example, readers interested in the methodologies can consult the introductory textbook [131], and the comprehensive textbooks [19, 81]. For grounding of machine learning in classic statistics, see [184]. For computational learning theory, see [97, 176] for the Vapnik-Chervonenkis (VC) dimension and Probably Approximately Correct (PAC) learning framework, and Chapter 4 in [153] for an introduction to the Rademacher complexity. For a perspective from information theory, see [119]. For a perspective that views machine learning as an important part of artificial intelligence, see [147].

CHAPTER 2

Overview of Semi-Supervised Learning

2.1 LEARNING FROM BOTH LABELED AND UNLABELED DATA

As the name suggests, *semi-supervised learning* is somewhere between unsupervised and supervised learning. In fact, most semi-supervised learning strategies are based on extending either unsupervised or supervised learning to include additional information typical of the other learning paradigm. Specifically, semi-supervised learning encompasses several different settings, including:

- *Semi-supervised classification*. Also known as classification with labeled and unlabeled data (or partially labeled data), this is an extension to the supervised classification problem. The training data consists of both l labeled instances $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ and u unlabeled instances $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. One typically assumes that there is much more unlabeled data than labeled data, i.e., $u \gg l$. The goal of semi-supervised classification is to train a classifier f from both the labeled and unlabeled data, such that it is better than the supervised classifier trained on the labeled data alone.

- *Constrained clustering*. This is an extension to unsupervised clustering. The training data consists of unlabeled instances $\{x_i\}_{j=1}^{n}$, as well as some "supervised information" about the clusters. For example, such information can be so-called *must-link* constraints, that two instances $\mathbf{x}_i, \mathbf{x}_j$ must be in the same cluster; and *cannot-link* constraints, that $\mathbf{x}_i, \mathbf{x}_j$ cannot be in the same cluster. One can also constrain the size of the clusters. The goal of constrained clustering is to obtain better clustering than the clustering from unlabeled data alone.

There are other semi-supervised learning settings, including regression with labeled and unlabeled data, dimensionality reduction with labeled instances whose reduced feature representation is given, and so on. This book will focus on semi-supervised classification.

The study of semi-supervised learning is motivated by two factors: its practical value in building better computer algorithms, and its theoretical value in understanding learning in machines and humans.

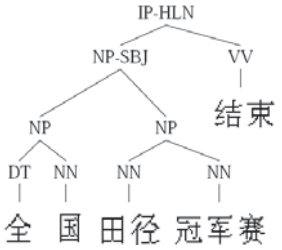
Semi-supervised learning has tremendous practical value. In many tasks, there is a paucity of labeled data. The labels y may be difficult to obtain because they require human annotators, special devices, or expensive and slow experiments. For example,

- In speech recognition, an instance $\mathbf{x}$ is a speech utterance, and the label y is the corresponding transcript. For example, here are some detailed phonetic transcripts of words as they are spoken:

film ⇒ `f ih_n uh_gl_n m`
be all ⇒ `bcl b iy iy_tr ao_tr ao l_dl`

Accurate transcription by human expert annotators can be extremely time consuming: it took as long as 400 hours to transcribe 1 hour of speech at the phonetic level for the Switchboard telephone conversational speech data [71] (recordings of randomly paired participants discussing various topics such as social, economic, political, and environmental issues).

- In natural language parsing, an instance $\mathbf{x}$ is a sentence, and the label y is the corresponding parse tree. An example parse tree for the Chinese sentence "The National Track and Field Championship has finished." is shown below.



The training data, consisting of (sentence, parse tree) pairs, is known as a treebank. Treebanks are time consuming to construct, and require the expertise of linguists: For a mere 4000 sentences in the Penn Chinese Treebank, experts took two years to manually create the corresponding parse trees.

- In spam filtering, an instance $\mathbf{x}$ is an email, and the label y is the user's judgment (spam or ham). In this situation, the bottleneck is an average user's patience to label a large number of emails.
- In video surveillance, an instance $\mathbf{x}$ is a video frame, and the label y is the identity of the object in the video. Manually labeling the objects in a large number of surveillance video frames is tedious and time consuming.
- In protein 3D structure prediction, an instance $\mathbf{x}$ is a DNA sequence, and the label y is the 3D protein folding structure. It can take months of expensive laboratory work by expert crystallographers to identify the 3D structure of a single protein.

While labeled data $(\mathbf{x}, y)$ is difficult to obtain in these domains, unlabeled data $\mathbf{x}$ is available in large quantity and easy to collect: speech utterances can be recorded from radio broadcasts; text sentences can be crawled from the World Wide Web; emails are sitting on the mail server; surveillance cameras run 24 hours a day; and DNA sequences of proteins are readily available from gene databases. However, traditional supervised learning methods cannot use unlabeled data in training classifiers.

Semi-supervised learning is attractive because it can potentially utilize both labeled and unlabeled data to achieve better performance than supervised learning. From a different perspective, semi-supervised learning may achieve the same level of performance as supervised learning, but with fewer labeled instances. This reduces the annotation effort, which leads to reduced cost. We will present several computational models in Chapters 3,4,5, 6.

Semi-supervised learning also provides a computational model of how humans learn from labeled and unlabeled data. Consider the task of concept learning in children, which is similar to classification: an instance $\mathbf{x}$ is an object (e.g., an animal), and the label y is the corresponding concept (e.g., dog). Young children receive labeled data from teachers (e.g., Daddy points to a brown animal and says "dog!"). But more often they observe various animals by themselves without receiving explicit labels. It seems self-evident that children are able to combine labeled and unlabeled data to facilitate concept learning. The study of semi-supervised learning is therefore an opportunity to bridge machine learning and human learning. We will discuss some recent studies in Chapter 7.

2.2 HOW IS SEMI-SUPERVISED LEARNING POSSIBLE?

At first glance, it might seem paradoxical that one can learn anything about a predictor $f : \mathcal{X} \mapsto \mathcal{Y}$ from unlabeled data. After all, f is about the mapping from instance $\mathbf{x}$ to label y, yet unlabeled data does not provide any examples of such a mapping. The answer lies in the assumptions one makes about the link between the distribution of unlabeled data $P(\mathbf{x})$ and the target label.

Figure 2.1 shows a simple example of semi-supervised learning. Let each instance be represented by a one-dimensional feature $x \in \mathbb{R}$. There are two classes: positive and negative. Consider the following two scenarios:

1. In supervised learning, we are given only two labeled training instances $(\mathbf{x}_1, y_1) = (-1, -)$ and $(\mathbf{x}_2, y_2) = (1, +)$, shown as the red and blue symbols in the figure, respectively. The best estimate of the decision boundary is obviously $\mathbf{x} = 0$: all instances with $\mathbf{x} < 0$ should be classified as $y = -$, while those with $\mathbf{x} \geq 0$ as $y = +$.

2. In addition, we are also given a large number of unlabeled instances, shown as green dots in the figure. The correct class labels for these unlabeled examples are unknown. However, we observe that they form two groups. *Under the assumption* that instances in each class form a coherent group (e.g., $p(\mathbf{x}|y)$ is a Gaussian distribution, such that the instances from each class center around a central mean), this unlabeled data gives us more information. Specifically, it seems that the two labeled instances are not the most prototypical examples for the classes. Our *semi-supervised* estimate of the decision boundary should be between the two groups instead, at $\mathbf{x} \approx 0.4$.

If our assumption is true, then using both labeled and unlabeled data gives us a more reliable estimate of the decision boundary. Intuitively, the distribution of unlabeled data helps to identify regions with the same label, and the few labeled data then provide the actual labels. In this book, we will introduce a few other commonly used semi-supervised learning assumptions.

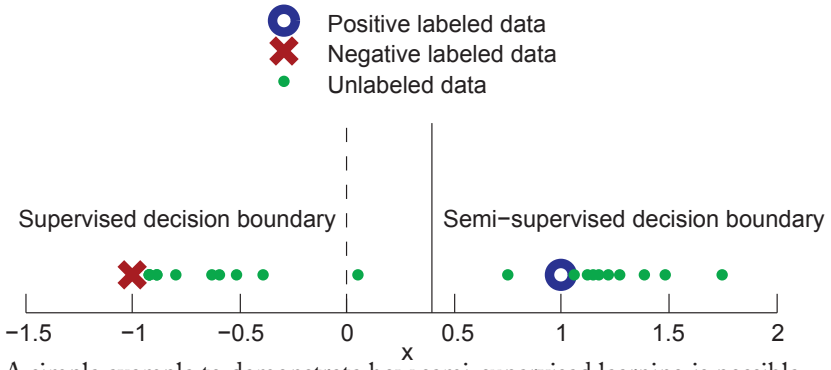


Figure 2.1: A simple example to demonstrate how semi-supervised learning is possible.

2.3 INDUCTIVE VS. TRANSDUCTIVE SEMI-SUPERVISED LEARNING

There are actually two slightly different semi-supervised learning settings, namely inductive and transductive semi-supervised learning. Recall that in supervised classification, the training sample is fully labeled, so one is always interested in the performance on future test data. In semi-supervised classification, however, the training sample contains some unlabeled data. Therefore, there are two distinct goals. One is to predict the labels on future test data. The other goal is to predict the labels on the unlabeled instances in the training sample. We call the former *inductive semi-supervised learning*, and the latter *transductive learning*.

Definition 2.1. Inductive semi-supervised learning. Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, inductive semi-supervised learning learns a function $f : \mathcal{X} \mapsto \mathcal{Y}$ so that f is expected to be a good predictor on future data, beyond $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.

Like in supervised learning, one can estimate the performance on future data by using a separate test sample $\{(\mathbf{x}_k, y_k)\}_{k=1}^{m}$, which is not available during training.

Definition 2.2. Transductive learning. Given a training sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$, transductive learning trains a function $f : \mathcal{X}^{l+u} \mapsto \mathcal{Y}^{l+u}$ so that f is expected to be a good predictor on the unlabeled data $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$. Note f is defined only on the given training sample, and is not required to make predictions outside. It is therefore a simpler function.

There is an interesting analogy: inductive semi-supervised learning is like an in-class exam, where the questions are not known in advance, and a student needs to prepare for all possible questions; in contrast, transductive learning is like a take-home exam, where the student knows the exam questions and needs not prepare beyond those.

2.4 CAVEATS

It seems reasonable that semi-supervised learning can use additional unlabeled data, which by itself does not carry information on the mapping $\mathcal{X} \mapsto \mathcal{Y}$, to learn a better predictor f. As mentioned earlier, the key lies in the semi-supervised model *assumptions* about the link between the marginal distribution $P(\mathbf{x})$ and the conditional distribution $P(y|\mathbf{x})$. There are several different semi-supervised learning methods, and each makes slightly different assumptions about this link. These methods include self-training, probabilistic generative models, co-training, graph-based models, semi-supervised support vector machines, and so on. In the next several chapters, we will go through these models and discuss their assumptions. In Section 8.2, we will also give some theoretic justification. Empirically, these semi-supervised learning models do produce better classifiers than supervised learning on some data sets.

However, it is worth pointing out that blindly selecting a semi-supervised learning method for a specific task will not necessarily improve performance over supervised learning. In fact, unlabeled data can lead to *worse* performance with the wrong link assumptions. The following example demonstrates this sensitivity to model assumptions by comparing supervised learning performance with several semi-supervised learning approaches on a simple classification problem. Don't worry if these approaches appear mysterious; we will explain how they work in detail in the rest of the book. For now, the main point is that semi-supervised learning performance depends on the correctness of the assumptions made by the model in question.

Example 2.3. Consider a classification task where there are two classes, each with a Gaussian distribution. The two Gaussian distributions heavily overlap (top panel of Figure 2.2). The true decision boundary lies in the middle of the two distributions, shown as a dotted line. Since we know the true distributions, we can compute test sample error rates based on the probability mass of each Gaussian that falls on the incorrect side of the decision boundary. Due to the overlapping class distributions, the optimal error rate (i.e., the Bayes error) is 21.2%.

For supervised learning, the learned decision boundary is in the middle of the two labeled instances, and the unlabeled instances are ignored. See, for example, the thick solid line in the second panel of Figure 2.2. We note that it is away from the true decision boundary, because the two labeled instances are randomly sampled. If we were to draw two other labeled instances, the learned decision boundary would change, but most likely would still be off (see other panels of Figure 2.2). On average, the *expected* learned decision boundary will coincide with the true boundary, but for any given draw of labeled data it will be off quite a bit. We say that the learned boundary has high variance. To evaluate supervised learning, and the semi-supervised learning methods introduced below, we drew 1000 training samples, each with one labeled and 99 unlabeled instances per class. In contrast to the optimal decision boundary, the decision boundaries found using supervised learning have an average test sample error rate of 31.6%. The average decision boundary lies at 0.02 (compared to the optimal boundary of 0), but has standard deviation of 0.72.

Now without presenting the details, we show the learned decision boundaries of three semi-supervised learning models on the training data. These models will be presented in detail in later

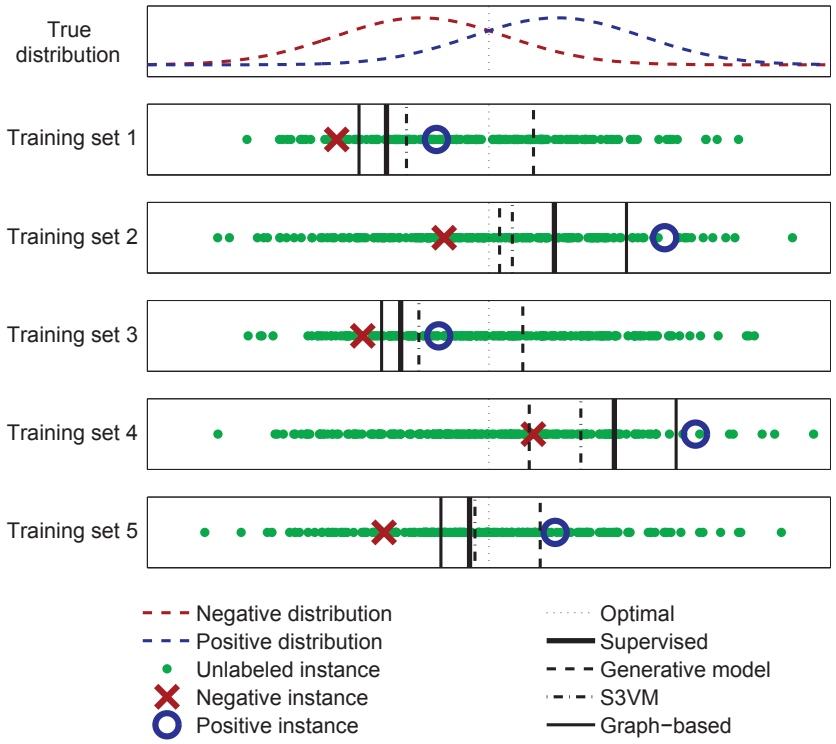


Figure 2.2: Two classes drawn from overlapping Gaussian distributions (top panel). Decision boundaries learned by several algorithms are shown for five random samples of labeled and unlabeled training samples.

chapters. The first one is a probabilistic generative model with two Gaussian distributions learned with EM (Chapter 3)—this model makes the correct model assumption. The decision boundaries are shown in Figure 2.2 as dashed lines. In this case, the boundaries tend to be closer to the true boundary and similar to one another, i.e., this algorithm has low variance. The 1000-trial average test sample error rate for this algorithm is 30.2%. The average decision boundary is at -0.003 with a standard deviation of 0.55, indicating the algorithm is both more accurate and more stable than the supervised model.

The second model is a semi-supervised support vector machine (Chapter 6), which assumes that the decision boundary should not pass through dense unlabeled data regions. However, since the two classes strongly overlap, the true decision boundary actually passes through the densest region. Therefore, the model assumption does not entirely match the task. The learned decision boundaries are shown in Figure 2.2 as dash-dotted lines.[1] The result is better than supervised classification and performs about the same as the probabilistic generative model that makes the correct model

[1]The semi-supervised support vector machine results were obtained using transductive SVM code similar to SVM-light.

assumption. The average test sample error rate here is 29.6%, with an average decision boundary of 0.01 (standard deviation 0.48). Despite the wrong model assumption, this approach uses knowledge that the two classes contain roughly the same number of instances, so the decision boundaries are drawn toward the center. This might explain the surprisingly good performance compared to the correct model.

The third approach is a graph-based model (Chapter 5), with a typical way to generate the graph: any two instances in the labeled and unlabeled data are connected by an edge. The edge weight is large if the two instances are close to each other, and small if they are far away. The model assumption is that instances connected with large-weight edges tend to have the same label. However, in this particular example where the two classes overlap, instances from different classes can be quite close and connected by large-weight edges. Therefore, the model assumption does not match the task either. The results using this model are shown in Figure 2.2 as thin solid lines.[2] The graph-based models' average test sample error rate is 36.4%, with an average decision boundary at 0.03 (standard deviation 1.23). The graph-based model is inappropriate for this task and performs even worse than supervised learning.

As the above example shows, the model assumption plays an important role in semi-supervised learning. It makes up for the lack of labeled data, and can determine the quality of the predictor. However, making the right assumptions (or detecting wrong assumptions) remains an open question in semi-supervised learning. This means the question "which semi-supervised model should I use?" does not have an easy answer. Consequently, this book will mainly present methodology. Most chapters will introduce a distinct family of semi-supervised learning models. We start with a simple semi-supervised classification model: self-training.

2.5 SELF-TRAINING MODELS

Self-training is characterized by the fact that the learning process uses its own predictions to teach itself. For this reason, it is also called self-teaching or bootstrapping (not to be confused with the statistical procedure with the same name). Self-training can be either inductive or transductive, depending on the nature of the predictor f.

Algorithm 2.4. Self-training.

Input: labeled data $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$, *unlabeled data* $\{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
1. Initially, let $L = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ *and* $U = \{\mathbf{x}_j\}_{j=l+1}^{l+u}$.
2. Repeat:
3. *Train* f *from* L *using supervised learning.*
4. *Apply* f *to the unlabeled instances in* U.

[2]The graph-based model used here featured a Gaussian-weighted graph ($w_{ij} = \exp \frac{||x_i - x_j||^2}{2\sigma^2}$, with $\sigma = 0.1$), and predictions were made using the closed-form harmonic function solution. While this is a transductive method, we calculate the boundary as the value on the x-axis where the predicted label changes.