Mastering machine learning chapter 3

Introduction to Semi-Supervised Learning

Semi-supervised learning is a machine learning branch that tries to solve problems that include both labeled and unlabeled data by employing both concepts clustering and classification methods.

The high availability of unlabeled and the dificullity of labeling datasets. drove many researchers to investigate samples to a larger unlabeled population, without loss of accuracy.

In this branch we will discuss the following:

- The semi supervised scenario
- The different approaches to semi-supervised learning.
- The assumptions needed to efficiently operate on such scenario.

We'll then move to several present semi-supervised algorithms
Example algorithms include:

- The Generative Gaussian Mixture algorithm
- Self-Training
- Co-training

Semi-supervised scenario

A typical semi supervised scenario is not very different from a supervised one
If we have Pdata

$$p_{data}(\bar{x},\bar{y}) = p(\bar{y}|\bar{x})p(\bar{x}) \text{ or } p(\bar{x}|\bar{y})p(\bar{y})$$

An important assumption about the unlabeled
samples is that their labels are supposed to be missing at random.
Without any correlation with the actual label distribution.

In some case it is impossible to use semi-supervised learning
If the unlabeled points are drawn from different distributions or from regions of P data excluded from the training process, the final result can be quite a lot worse.

cross-validation and comparisons are the best practices to employ when evaluating a scenario.

in a semi-supervised scenario, we are often forced to model p(X(hat))
in order to exploit the unlabeled samples.

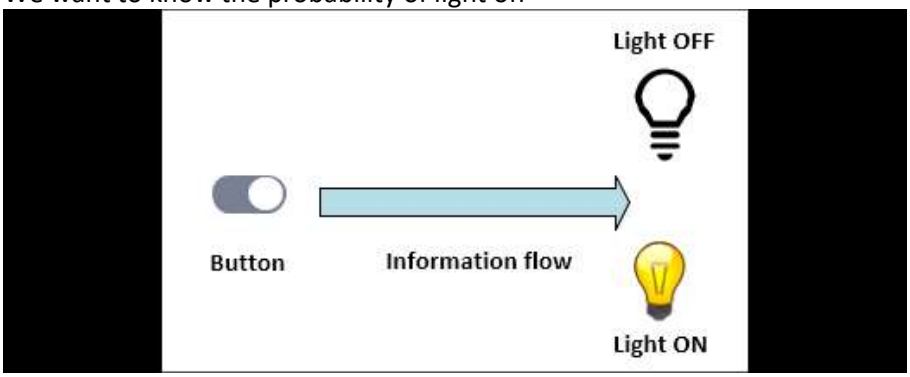
Types of scenarios:

Causal scenarios:

We suppose that the knowledge of increases p(x(hat)) the knowledge of p(y|X) ?
The distribution of the effects, given some causes, is independent of the

distribution of the causes.

We want to know the probability of light on



knowing that the probability of a frequency range isp0 has no right to influence our knowledge of p(light|button = ON).

we are in fact modeling p(cause | effect) (instead of p(effect|cause)) and, clearly, the knowledge of p(effect) can affect the conditional probability.

since the light is triggered by the button, if we have carried out N experiments, we can create an initial model p(button =ON | light), but we're not sure that we are taking into account all possible outcomes (light on/off). Therefore in our first estimation, we are likely to introduce an error.

The knowledge of p(light) can help us reduce such an error because we know that, whenever the probability to observe the light on is large, the probability of a correct button frequency is also large.

(note: if the probability of destruction of trees is high given lightning than
Than the probability of have lightning frequency is also large).
We often consider the class(target) as the cause and the attributes as
the effects
(think : the class as the features and features as the class)

(note: check every time that the model improves more than the supervised model.)

Transductive learning:

When a semi-supervised model is aimed at finding the labels for the unlabeled samples, the approach is called transductive learning.

In this case, we're not interested in modeling the whole distribution p(y(hat)|X(hat)) ,which implies determining the density of both datasets, but rather in finding p(y|X) only for the unlabeled points.

This is used as:

- time-saving
- our goal is more oriented at improving our knowledge about the unlabeled dataset

this scenario implies that the knowledge of p(x(hat)) can improve our knowledge of p(y(hat)|X(hat)); it's not suitable for purely causal processes
(we use in Transductive learning the unlabeled data is used as test set).

Inductive Learning.

Contrary to transductive learning, inductive learning considers all the X data points, and tries to determine a complete. p(y(hat)|X(hat)) or a function y=f(X(hat)) that can map both labeled and unlabeled points to their corresponding labels.

this method is more complex and requires more computational time.
according to Vapnik's principle – that we shouldn't solve a more general problem as an intermediate step in solving a specific problem– if it's not required or necessary, it's always better to pick the most pragmatic solution and, possibly, expand it if the problem requires further details.

Semi-supervised assumptions:

A semi-supervised learning is not guaranteed to improve a supervised model. A wrong choice could lead to a dramatic worsening in performance. However, it's possible to state some fundamental assumptions that are required for semi-supervised learning to work properly. They're not always mathematically proven theorems, but rather empirical observations that justify.

Types of assumptions:

Smoothness assumption:

Let's consider a real-valued function f(x) and the corresponding metric spaces X and Y. Such a function is said to be Lipschitz-continuous if:

$$\exists K : \forall x_1, x_2 \in X \Rightarrow d_Y\big(f(x_1), f(x_2)\big) \leq K d_X(x_1, x_2)$$

if two points x1 and x2 are near, the corresponding output values y1 and y2 cannot be arbitrarily far from each other. This condition is fundamental in regression problems.

if we need to predict the output for a point xt : x1 < xt < x2 and the regressor is Lipschitz-continuous, we can be sure that yt will be correctly bounded by y1 and y2. This behavior is called general smoothness.

if two points are in a high-density region (cluster) and they are close, then the corresponding outputs must be close too.

This can be expressed as :

$$if\ f(\bar{x}_c; \bar{\theta}) = y_c, \exists\, \delta > 0 : \forall\, \bar{x} \in X : d(\bar{x}, \bar{x}_c) < \delta \Rightarrow f(\bar{x}; \bar{\theta}) = y_c$$

if two samples are in a low-density region they can belong to different clusters and their labels can be very different. This isn't always true.

Cluster Assumption:

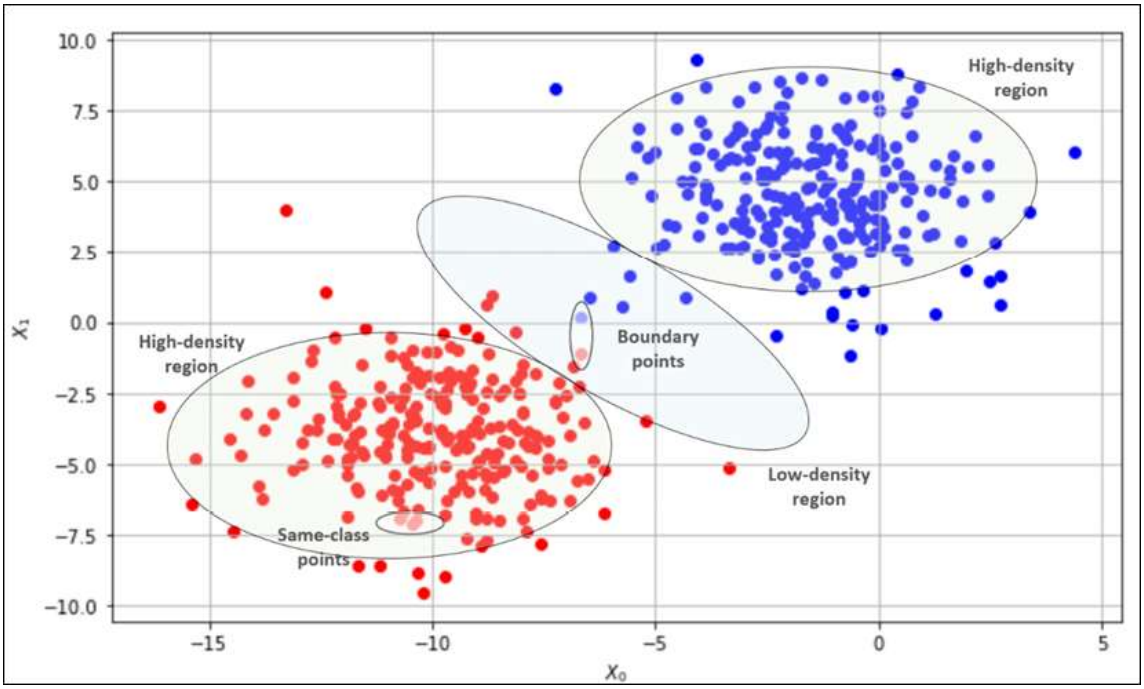
This assumption is strictly linked to the previous one. Clusters are high-density regions;

therefore if two points are close, they're likely to belong to the same cluster and their labels must be the same.

Low-density regions are separation spaces; therefore samples with low density regions are likely to be boundary points and their classes can be different

To understand:
Think of SVM only the support vectors should be in low density regions

Let's consider bidimensional example:



In a semi supervised scenario we don't know the label of a point belonging to a high-density However if it is close enough to a labeled point where it is possible to where it is possible to Build a ball where all the points have the same average density then we're allowed to predict The label of our test sample.

Low density separation will Discussed in the following chapter

Manifold assumption:

The least intuitive example but can be the extremely useful to reduce the complexity of Many problems.

Definition of n-manifold:
An n-manifold is a topological space that is globally curved, but locally homeomorphic to n-dimensional Euclidean space.

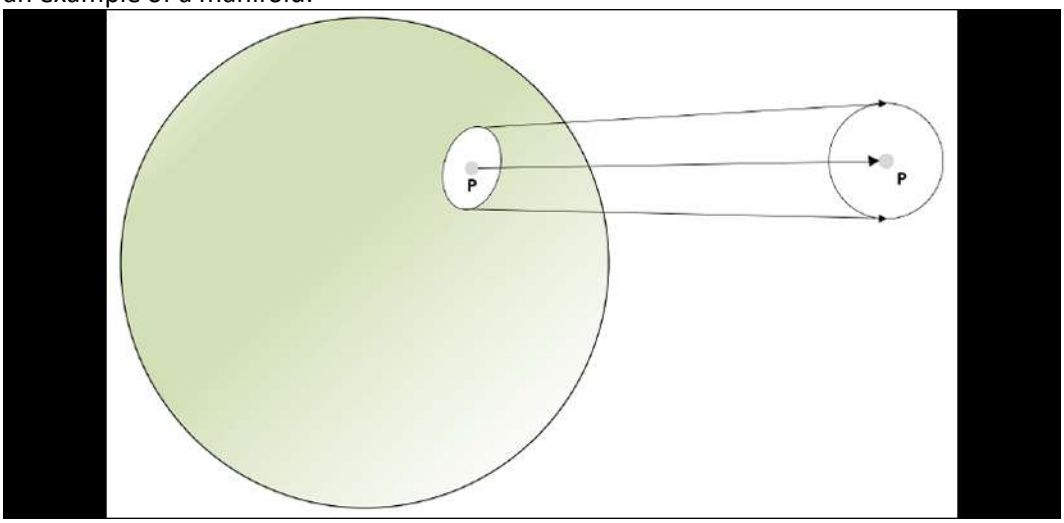
In other words it's possible to select a sufficiently small region and deform it into a standard Euclidean flat space, But this is not true when considering the whole space.

For example Earth from space we might see

In other words it's possible to select a sufficiently small region and deform it into a standard Euclidean flat space, But this is not true when considering the whole space.

For example Earth from space, we might see It inhabitants are uniformly distributed over the whole volume. We know that this is false, in fact We can create maps which are represented on 2 dim manifolds and it doesn't make sense to humans on 3 dim vectors.

an example of a manifold:



if we have N 1000-dimensional bounded vectors, they are enclosed into a 1000-dimensional hypercube with edge-length equal to r. The corresponding n-volume is r^p = r^1000; therefore, the probability of filling the entire space is very small (and decreases with p). What we observe, instead, is a high density on a lower dimensional manifold.

This assumption authorizes us to apply dimensionality reduction methods in order to avoid the Curse of Dimensionality.

when the dimensionality of the samples increases, in order to achieve high accuracy, it's necessary to use more and more samples.

However the accuracy of statistical classifiers is inversely proportional to the dimensionality of the samples.
This means that whenever it's possible to work on lower dimensional manifolds (in particular, in semi-supervised scenarios), two advantages are achieved:

- Less computational time and memory consumption
- Larger classification accuracy

Imagine if I have:
2,000 × 1,000 RGB canvas. Each pixel is encoded with 24 bits, so it can have 224 = 16,777,216 possible values. If all the pixels in the canvas (2,000,000) are independent, the total number of images is 16,777,2162000000. It's not difficult to understand that this number is extremely huge

all the following examples that involve random numbers, the seed is set to (np.random.seed(1000)).