Computational Learning Theory **Spring Semester, 2002**

Lecture 1: March 14, 2002

Lecturer: Yishay Mansour *Scribe: Irit Gat-Viks, Giora Unger, Dotan Emanuel*[1]

1.1 Introduction and Overview

1.1.1 What is machine learning?

The goal of machine learning is to develop computer programs that automatically improve with experience. Such programs adapt themselves to changing conditions and extract meaningful information from raw data.

Typical applications

There are numerous applications of machine learning. Here are some examples:

- Classification/Clustering problems: data mining, information Retrieval, web search.
- Self-customization: customizing e-mail programs, newsgroup reading programs.
- Control: Robots, Dialog system.
- Complicated software: driving a car, playing a game (backgammon).

Why now ?

There are several reasons for machine learning to become a hot field during the past 3-4 years. The technology rapidly advances, in both the algorithmic and the theoretic fields. In addition, there is a lot of information (online data sources) and computational power available.

Finally, and probably most importantly, the consumers need complex tasks to be performed more efficiently. That can be achieved by learning machines.

[1] Partially based on scribes written by Sigal Sahar (February 19, 1997), and by Anat Golan and Guy Eden (March 6, 1996).

A few examples:

Example 1: Credit Risk Analysis.

Consider a bank, that would like to grant credit only to "good" customers. The bank has a lot of information about its customers, such as age, income, marital status, financial reputation, etc. When a customer requests a line of credit, the bank must decide whether or not to approve such a request. Traditionally, the criteria for granting or denying credit are set by human beings.

Can human beings perform this task more reliably and efficiently than a machine? Not necessarily. One can think about a program, that would receive, as input, the data about the bank's hundred of thousands customers, and will output a set of decision rules that will describe the "Good" customers, the one who have proven to be worth of a credit line.

Example 2: News clustering.

Given a source of news (e.g.: Reuters), the goal is to achieve a basic category classification (e.g.: business, sports, politics). A human will do it much better then a machine, but it is a laborious task. We would like to be able to supply the machine with a large set of documents from each category and let it learn the classification rules, or even let it specify the categories and sub-categories. Notice that the interesting (and difficult) task is not the mere classification of articles into specified categories, but to learn which categories and subcategories actually exist.

Example 3: Robot control.

The goal here is to control a robot in an unknown environment. Unlike the previous two examples, we deal with more sophisticated set of problems. The machine needs not only to perform pre-defined tasks, but also to explore new places and actions, in order to use the acquired knowledge for gaining additional benefits.

For example, consider a robot whose task is to touch as many balls as possible. However, the locations of the balls are unknown to the robot. Hence, before actually approaching a ball, the robot must find it!

This new aspect also gives rise to a new type of bugs in machine learning programs - failure to efficiently explore new actions.

Types of models

Generally speaking, there are three types of learning models:

- Supervised learning - the machine is granted with access to classified data that includes the decision parameters and the right classification. In this case it can build a model, verify it against the given classification so the machine can "learn" the right classification rules.

- Unsupervised learning - the machine is given access to data and has no knowledge on how the data should be classified. In this case it can try to find internal data dependencies.

- Control learning - the machine selects actions and observes feedbacks, as in the robot control example above. The goal is to maximize the long-term cumulative return.

Related disciplines

There are several disciplines related to machine learning, such as: Computational learning theory, Artificial intelligence, Statistics, Control theory, Information Theory, Philosophy, and Psychology & neurobiology.

In this course, we'll concentrate on Computational Learning Theory, and will also consider some aspects of Statistics and Artificial intelligence.

A glimpse into the future

Today we have the first-generation algorithms, used in fields of information retrieval, web-search, voice recognition and others. Many more application are yet to come.

The flexibility and benefits that can be gained using a learning machine include creation of well-formed widely available data-bases. Increasing of computational power would probably help addressing more problems in the new fields, such as networking and security control.

1.1.2 Complete information

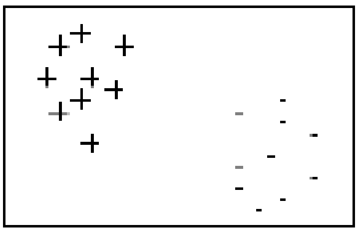

Figure 1.1: Classified data set

In a two dimensional space (two coordinates) we are given a data set that has been classified into two groups: '+' and '-' as in Figure 1.1. We would like to be able to classify a new point, (x,y).

In this scenario, we assume that we are provided with **Complete Information**: the distribution that generates the $'+'$s is D_+, the distribution that generates the $'-'$s is D_- , and the samples are drawn from the distributions with equal probability $D(z) = \frac{1}{2}D_+(z) + \frac{1}{2}D_-$ (z). Given a point (x, y) We need to calculate $Pr[+|(x, y)]$ and $Pr[-|(x, y)]$ (note that their sum is one).

$$Pr[+|(x,y)] = \frac{Pr[(x,y)|+] \cdot Pr[+]}{Pr[(x,y)]} = \frac{D_+(x,y) \cdot \frac{1}{2}}{\frac{1}{2}D_+(x,y) + \frac{1}{2}D_-(x,y)} = \alpha$$

We are now interested in classifying the point (x,y). Which classification shall we choose? The obvious choice would be $'+'$ if D_+ is more likely then D_- (meaning $\alpha > \frac{1}{2}$) but this approach is not always the best one.
Assume we are dealing with a medical test for cancer where $'+'$ means the patient has cancer, $'-'$ means she doesn't. Here we can allow ourselves to do the mistake of telling someone without cancer that she's probably ill (and sending her to further tests) but not vice-versa. If our classification depends on the "loss" we will experience for every misclassification a huge compensation lawsuit. On the other hand the cost of some extra tests is negligible. We, of course, want to minimize this loss. So lets look at some loss models.

Prediction and loss model

Boolean error
If we commit an error, we lose 1. Otherwise, we lose nothing. We compare the
$Pr[+|(x, y)] = p$ (The probability of receiving '+' sign in the point x,y) versus
$Pr[-|(x, y)] = 1 - p$ (The probability of receiving '+' sign in the point x,y)
and choose the higher value. (if $p > \frac{1}{2}$ we guess '+', if $p < \frac{1}{2}$ we guess '-').
Using this loss function we do not recover the probabilities in any way, $p = 0.99$ or $p = 0.51$ would both result in choosing '+'.

Quadratic Loss
Here we choose a "real number" q for the outcome of 1.
The loss is given by $(p - q)^2$ for an outcome of 1, and $((1 - p) \cdot (1 - q))^2$ for an outcome of 0. and the total expected loss is $(p - q)^2$
The minimal loss will be achieved for $p = q$.
Using this loss function we must know p in order to approximate the loss.

Logarithmic Loss
Here we also choose a "real number" q. We'll be penalized as follows:

- $-\log q$ for an erroneous '+'
- $-\log(1 - q)$ for an erroneous '-'

Our total expected loss will be $-p \cdot \log q - (1-p) \cdot \log(1-q)$. Then $p = q$ we will achieve minimal loss (one can see this by equating the first derivative to zero). Therefore, one can think of q as the estimate for p. Using this loss function we do not need to know p.

1.1.3 The basic PAC Model

The problem:
We have an unknown distribution D over domain X, an unknown target function $f(x)$ that classifies each point $x \in X$, and a set of hypotheses H.

The Goal:
We are looking for a hypothesis $h \in H$ that minimizes

$$Pr_D[h(x) \neq f(x)]$$

In other words we want to minimize the *true error*

$$\varepsilon(h) = Pr_D[h(x) \neq f(x)]$$

(Note: ε may be positive since we are not guaranteed that f(x) itself is contained in H).

The twist:
Had we known D, we could simply go over all $h \in H$ and find the hypothesis that minimizes $\varepsilon(h)$. However, we do not have complete information, but a *sample* S of m examples drawn according to i.i.d from D.
Thus, we can minimize only the **observed error**:

$$\varepsilon'(h) = \frac{1}{m}[x \in S | h(x) \neq f(x)]$$

(the error on the samples).
It means we may find the best hypothesis that matches f(x) *over the sample*, but not necessary the one that matches f(x) over D. This raises a basic question: "How close is $\varepsilon(h)$ to $\varepsilon'(h)$? Is the best hypothesis on the sample is also the best on the entire distribution D?"

Bayesian Theory

We start out with a family of hypotheses that describe the distribution of the '+'s and the '-'s, D_+ and D_-. According to a preliminary distribution of hypotheses, we will choose the hypothesis for the samples. For example, if the distributions are normal we have $\frac{1}{2}N_- + \frac{1}{2}N_+$.

Given a sample set S=$< (x_i, y_i), b_i >$ where $b_i \in \{0,1\}$, we want to construct the distribution of the hypotheses after the sampling:

$$Pr[h|S] = \frac{Pr[S|h] \cdot Pr[h]}{Pr[S]}$$

Now we have the distribution Pr$[h|S]$ that defines the probability of each hypothesis given the sample. Given a new point we can calculate the probability it will be classified '+' or '-'. Please note that this method is not practical since one needs to go over all the hypotheses and for each one calculate its distribution.
We can use three types of hypothesis in Bayesian Theory:

- Maximum Likelihood (ML):
$$Pr[S|h]$$
- Maximum a Posteriori (MAP):
$$Pr[h|S]$$
- Bayesian Predictor:
$$\sum h(x) \cdot Pr[h|S]$$

Nearest Neighbor Methods

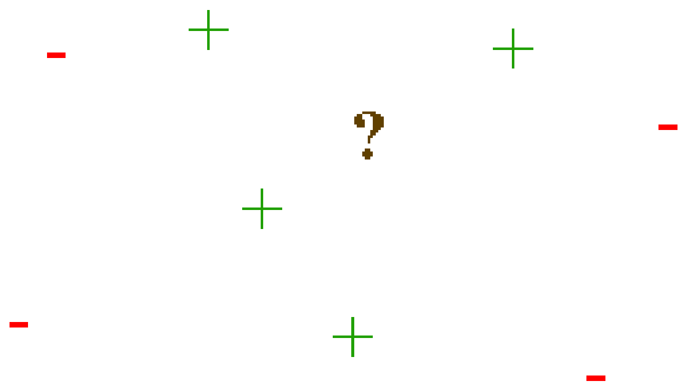

Figure 1.2: Which is the nearest neighbor?

We receive a training set $((x_i, y_i), b_i)$ where (x_i, y_i) is a point in our problem space and b_i is the classification. Given a new point(x,y) we can find the "closest" point in our training set and classify (x,y) the same way. This method is called 1-NN.

We can also find the K closest neighbors of (x,y) and classify according to the result of a majority vote of the K-nearest neighbors. This method is (surprisingly...) called K-NN.

The nearest neighbors approach assumes we have a structured space and a metric defined on that space. When the dimensionality of the problem increases, almost all the points appear uniformly distributed over the entire space and the method becomes problematic.

1.1.4 Computational Methods

We will now describe a different method for finding an $h \in H$ with low observed error, and a heuristic algorithm for some specific classes. In most of the cases computational tasks are provably hard.

Separating Hyperplane

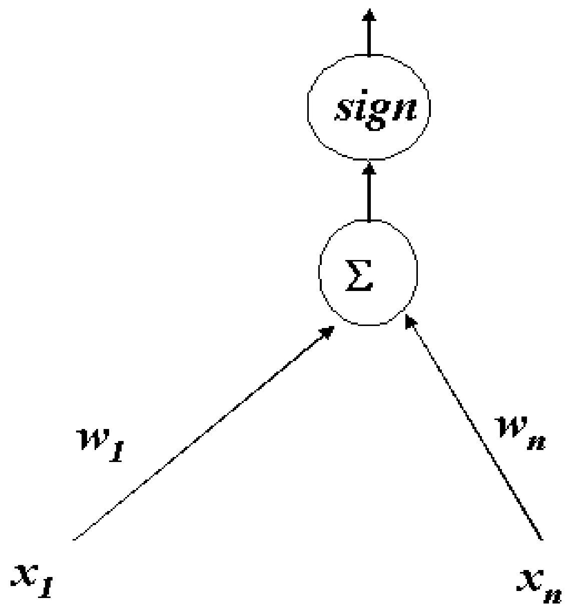

Figure 1.3: A perceptron

We are looking for a hyperplane that would separate the problem space into two different classification regions. We define the Hyperplane using a simple linear operator called peceptron; Perceptron=$sign(\sum_1^n x_i w_i)$, where:

- x_i are the input, and $w_1 ... w_n$ are the input weights
- we need to find $w_1 ... w_n$, that would classify the input correctly, that is - would minimize the error.

A hyperplane is a very simple model, but also a limited one.

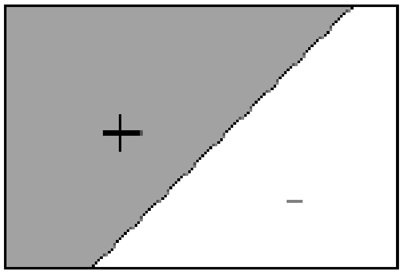

Figure 1.4: A separating hyperplane in two-dimensional space

Neural Networks

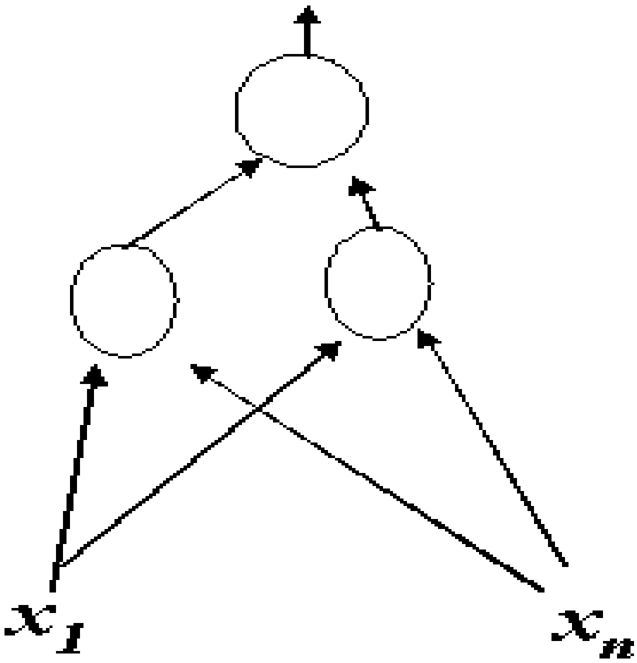

Figure 1.5: A neural network

A neural network (feed forward network) is a combination of a few layers of Perceptron. The layers allow us to define a more complex classification regions using the intersection of many hyperplanes. The neurons in the net will be activated using a Sigmoidal gates and not a sign threshold function like in the Perceptron. We will need a continuous function for the

gate in order to apply a learning algorithm called *Back Propagation*.
We define the neuron output to be

$$N_i = \frac{1}{(1 + e^{-a})}$$

where

$$a = \sum x_i w_i$$

Neural networks are much stronger model then a single Perceptron. They are used in many applications.

Decision Trees

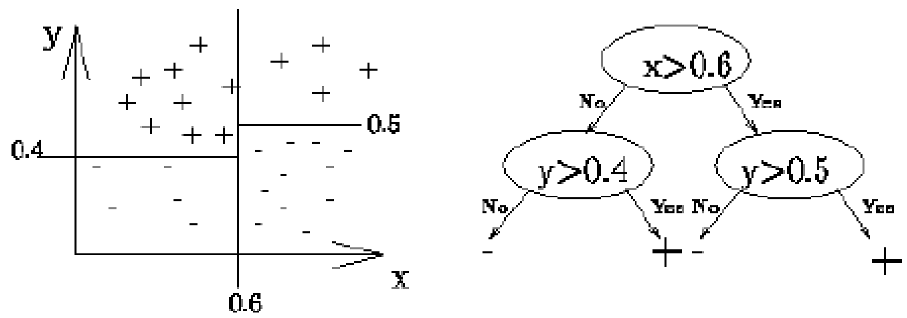

Figure 1.6: An example with a decision tree

The question is, given a sample set, how does one construct the decision tree? We need:

- A criterion for splitting the space (the questions at the nodes). The criterion is used recursively for the tree construction.
- A criterion to determine how much we should allow the tree to grow (the class complexity).

Our aim is to find a small decision tree with small observed error. this will be done in two phases:

PHASE I:
Construct the tree greedily, using a local index function. Ginni Index : G(x) = x(1-x), Entropy H(x) etc.

PHASE II: Prune the decision tree while maintaining low observed error.

1.1.5 Complexity versus Generalization

When choosing a model we have a permanent tradeoff of hypothesis complexity versus observed error. More complex hypothesis have lower observed error, but might have higher true error.
A complex hypothesis that has many degrees of freedom would use all of then in order to reduce the observed error. By doing this, it will adapt to noise or to the specific sample structure and not to the structure of the true underlying distribution.
In order to avoid the "over-fitting" problem we would like to limit the complexity of the model and try to find a simple model even if it has a slightly higher observed error.

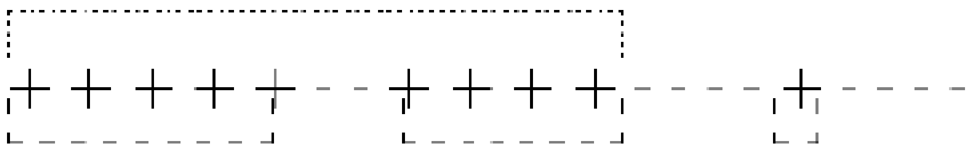

Figure 1.7: Two hypotheses for a given sample set. The dotted (upper) hypotheses is much better then the lower one, Though it's observed error on the sample data is higher

When we select a model we'd like to penalize the more complex model (just for being complex) and compensate the simple one for its possibly higher observed error. We will do that by using one of the following criteria:

- *Minimum Description Length*

$$\varepsilon' + |code\ length\ of\ h|$$

- *Structural Risk Minimization*

$$\varepsilon' + \sqrt{\frac{\log |h|}{m}}$$

(Where m is the number of items in the sample)

General PAC Methodology

In the general PAC methodology we look for ways to:

- Minimize the observed error
- Search for a small size classifier

We may use some hand-tailored search methods for specific classes of problems.

Genetic algorithms

Genetic algorithms are search algorithms that get their inspiration from the genetic process in the biological world. Genetic algorithms perform "Local mutation" operations, "Cross-over" operations, and "survival of the fittest" of candidates. These operation are similar to search operations preformed on a search-tree.

Weak Learning

Weak learning is the process of taking predictions which are "weak" (predict slightly better then just tossing a coin) and build strong predictions (which give good estimation most of the time) out of them .
In weak learning we have a small class of predicates and a weak learning rule.

Weak learning Rule: Assume that for any distribution D, there is some predicate $h \in H$ that predicts better than $\frac{1}{2} + \varepsilon$.

We will now transform this weak learning into strong learning by using the boosting algorithm.
Boosting algorithm
Functions: a weighted majority of predicates.
Method: In order to create different predictions we will change the distribution of the sample set to target hard examples. The weight of an example will be exponential in the number of incorrect classifications.
since we have the weak learning assumption we can get a weak predicate for every distribution. In order to create different distributions without re-sampling we will change the weight we give to each item in the sample, thus artificially creating a new distribution.
This algorithm produces extremely good experimental results.

Support vector machine

So far we tried to decrease the dimension of the problem in order to provide simple models. In Support vector machine we would like to do the opposite! We are going to increase the dimension of the problem, and to project the problem to a higher dimension space. In that space we will use hyperplane hypothesis in order to classify the data.
We will look for a hyperplane that separates the data in the best way, i.e., has the maximum gap between the hyperplanes and the closest sample points from both sides.

1.1.6 Other Models

Fourier Transform

We will look at the data as if it was a signal, apply the fourier transform and then look at the obtained coefficients. We will focus on the large coefficients, which are above a specific threshold, to represent the regularity of the data set.

Reinforcement Learning

This model is used mainly for control problems. Changing the parameters also changes the behavior. This learning model is based on immediate feedback.

Unsupervised learning: Clustering

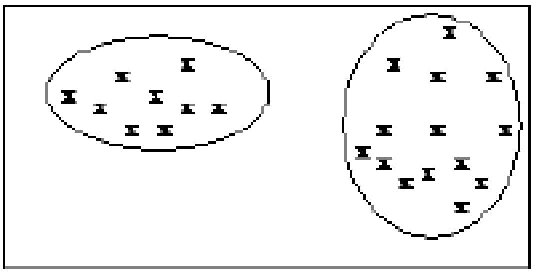

Figure 1.8: Clustering

In this case we do not have the classification of the sample set, but rather determine it ourselves. The idea is to find a "natural" division of the points in the sample set, as shown in Figure 1.8. Ideally, the subdivision will form a group for every possible classification.

1.1.7 Basic Concepts in Probability

During the course we'll often use probabilistic arguments. We recall here a few basic tools from probabilistic theory:

Markov Inequality

Let x be a non negative random variable (r.v.) and $E[x]$ the *expected value* (mean) of x, then:

$$Pr[x \geq \alpha] \quad \leq \quad \frac{E[x]}{\alpha}$$

Chebyshev Inequality

Suppose that x is arbitrary with variance $Var(x)$, then:

$$Pr[|x - \eta| \geq \beta] \leq \frac{Var(x)}{\beta^2} \quad where\ \eta = E[x].$$

Chernoff Inequality

Let the sequence of random variables $x_1, \ldots, x_n$ where $x_i \in \{0, 1\}$ be independent identically distributed (i.i.d.), and $Pr[x_i = 1] = p$, then:

$$Pr[|\sum_{i=1}^{n} \frac{x_i}{n} - p| \geq \lambda] \leq 2e^{-2\lambda^2 n}$$

The last inequality is especially interesting, since it gives us a tool to argue "how fast" the "observed mean" would converge to the "true mean".