Computational Learning Theory **Fall Semester, 2010**

Lecture 1: October 17, 2010

Lecturer: Yishay Mansour
Scribe: Aviv Gruber, Shai Hertz, Chavatzelet Tryster, Rinat Pichker[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. These applications are divided into several categories:

- **Classification problems**
  These problems are characterized by the fact that in the end there is only one answer: yes or no. For example spam detection - "Was this email a spam or not?" Another example is stock market prediction - "Was the prediction right or wrong?"
  Other examples are: search and fraud detection.
  Classification uses historical data in order to predict future instances.

- **Clustering**
  In these problems we wish to automatically divide the data into meaningful data groups. Usually there is more than one way of dividing the data into groups, therefore there isn't a right or wrong answer.
  Examples: document classification, news classification.
  It is possible for an outside user to tag the documents or news items, but we would like the tagging to be done automatically without requiring a human operator to tag every item.

- **Control**
  In these problems the learning algorithm controls what will happen. Examples: Robotics, Games.

[1]Partially based on scribes written by Irit Gat-Viks, Giora Unger, and Dotan Emanuel (March 14, 2002)

- **Collaborative Filtering**
  In these problems we wish to filter information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc.
  Examples: computer vision, speech recognition, translation, friend recommendation (facebook).

We will concentrate on the classification section. Most of the applications are relatively new applications.

Types of Machine Learning

Supervised Vs. unsupervised

Supervised learning means that the machine is granted 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. classification problems are an example of supervised learning.
Example of Supervised learning - A dataset with information regarding the financial details of different people, when they received a loan and whether the loan was returned. Based on this information we will try to predict whether other people, given their financial details, will return a loan.
Unsupervised learning - the machine is given access to data and has no knowledge of how the data should be classified. In this case it can try to find internal data dependencies, but there is no way to know whether the machine is right or wrong. Examples for unsupervised learning:

- Anomaly Detection: detecting an abnormal behavior in a system.
- Data Mining: the process of extracting an unknown pattern from the given data.

Active Vs. Passive

Passive learning means that the machine relies on a set of examples from the past and builds a function that will predict the future.
Active learning is characterized by the ability to create a new example and give it's classification.
For example: based on information about other customers we will build a profile of a new customer and ask the machine to classify its behavior.

Teacher Learning

Teacher learning means the machine tries to mimic an expert and classify examples like he does. An implementation of teacher learning are Expert Systems. An expert system is a

software that attempts to provide an answer to a problem, or clarify uncertainties where normally one or more human experts would need to be consulted.

Online Vs. Batch

Batch learning means that the machine gets a collection of examples and classifies them. This operation is repeated only once.
Online learning means that the machine gets one example at a time, classifies it, learns from its own answer, and repeats this process with each additional example.
In this course we will concentrate on supervised learning.
Most of the course will deal with the passive model, we will not assume a teacher and we will address both batch and online models.

Why do we need Machine Learning?

1. Tasks in which it is difficult to define precisely how they should be executed.
   For example driving a car - when you drive a car you use your human generalization capability - we would like the programs to act in the same way as the driver. Many times it's much simpler to give examples rather then rules for how to act (drive).

2. Tasks that are beyond human capability.
   For example: the analysis of huge databases, discovering interesting and unknown phenomenons.

In both cases writing a program that will solve these problems is difficult and standard programming does not provide tools to deal with these difficulties.

1.1.2 Building a machine learning model

In order to build a machine learning model we will have to answer the following questions:

1. We need to hypothesize how the examples (including the classification) are created. This is an assumption regarding the environment.

2. We need to decide how to compare the classifications of different machines. Obviously we would like to minimize the number of errors, but perhaps many small errors are preferable to a few big errors.

3. We need to introduce assumptions about the mapping between an example and its classification.

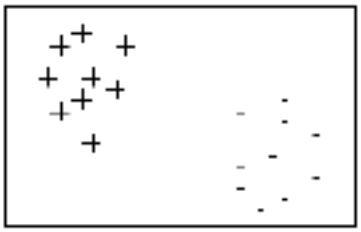

Figure 1.1: Classified data set

An example for a machine learning model

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).
We will assume that each example is a vector of the form $< 0, 1, 0, ..., 0 >$ and each vector has a classification of + or -.
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_- . Neither of the distributions are 100% certain. We will assume that 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 $Pr[-|(x, y)]$.

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

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.
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 make the mistake of telling someone without cancer that she's probably ill (and sending her for further tests) but not vice-versa. If our classification depends on the "loss", every misclassification will incur 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 (0-1 Loss)
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 $(1-q)^2$ for an outcome of 1, and q^2 for an outcome of 0. The total expected loss is $p \cdot (1-q)^2 + (1-p) \cdot q^2$
To find the minimum, we take the derivative:

$$2q(1-p) - 2p(1-q) = 2q - 2p = 0$$

The minimal loss will be achieved for $p = q$.
Using the quadratic loss function we minimize the loss by setting $p = q$.

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.

Note the difference between the quadratic loss and the logarithmic loss. Although both result in the choice of $p = q$, in the logarithmic loss it is possible to experience an infinite loss, when setting $q = 0$ or $q = 1$, which will make it less likely to give a such q. The quadratic loss will result in the loss of at most 1 for any q.

1.1.3 Assumptions of the classificaion

An explicit assumption

$$f(x) = \sum_{i=1}^{d} \alpha_i x_i$$

i.e. $f(x)$ is a linear function of the input. In this case, the algorithm will search for the α_i. If our hypothesis is correct and there are enough equations, we will be able to find all the

α_i. If the hypothesis is incorrect we will either get an incorrect value for a given α_i or there will be no solution, which will contradict our assumption.
A preferable assumption is dependent on how close is the classification to our example. I.e. the closer the classification to linear, the more precise the prediction.

1.1.4 Two basic approaches to the problem

1. Bayesian Inference

The Baysian inference assumes a prior distribution that contains all our beliefs as to how things should be. For instance it will contain the probability of the distribution being uniform, or gaussian, and if it is gaussian, what is the probability of the parameter being a certain value. The distribution will be relevant to both the examples and the classification function $f(x)$. Given a set S containing a set of points (vectors in a D dimensional plane): $S = < x_i, b_i >$ and $x_i \in X, b_i \in \{0, 1\}$ We would like to calculate:

1. $Pr[f(x) = 1|x, S] = \sum_{h \in H} h(x) Pr[f = h|S]$
2. $Pr[f = h|S]$

$Pr[f(x) = 1|S, x] = \sum_{h \in H} h(x) Pr[f = h|S]$ - Summing over all functions $h \in H$ the probability of the function h, given the set S.
$Pr[f = h|S] = \frac{Pr[S|f=h] \cdot Pr(f=h)}{Pr[S]}$ where $Pr[f = h]$ is given according to the prior distribution.
$Pr[S|f = h]$ denotes the probability of seeing the set S if h is indeed the correct classifier.
$Pr[S]$ is independent of the classifier and therefore often acts as a normalizer and can generally be ignored.
Other two popular Bayesian methods are:

- Maximum Likelihood (ML)
  In this method we wish to maximize the probability of seeing the set S given the function h. $Max_{h \in H} Pr[S|h]$. The prior distribution is all but ignored here.
- Maximum a posteriori (MAP)
  In this method we wish to maximize the probability of the function being h, given the set S. $Max_{h \in H} Pr[h|S]$

These two methods are identical when the distribution is uniform.

2. The basic PAC Model

The problem:
We have an unknown distribution D over domain X, set of samples S i.i.d., 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 the loss.

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

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**:

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

(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 $\hat{\varepsilon}(h)$? Is the best hypothesis on the sample also the best on the entire distribution D?" Another issue is the computational one - finding h that will minimize the observed error in an efficient manner.

1.1.5 Hypothesis Classes

Hyperplane

We are looking for a hyperplane that would separate the problem space into two different classification regions.
A hyperplane is a very simple model, but also a limited one. (See figure 1.2)

Decision Trees

(See Figure 1.3)

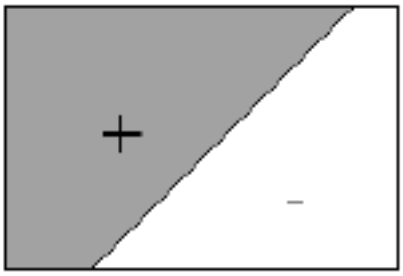

Figure 1.2: A separating hyperplane in two-dimensional space

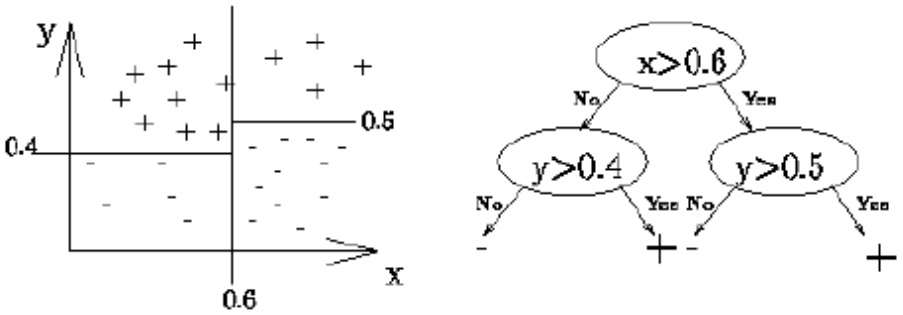

Figure 1.3: An example with a decision tree

1.1.6 General PAC Methodology

1: Choosing $h \in H$ that will minimize the observed error.
2: Proving the generalization error.

It is quite possible that we will achieve 'overfitting' because we are fitting the hypothesis to the data.

1.1.7 example

$X = \{0,1\}^d$ $f(x) \sim Random$ $H = \{x_1, x_2, ..., x_d\}$ - d functions, each one refers to a different attribute. $D \sim Uniform\{0,1\}^2$ What were to happen if we were to sample $\frac{1}{2}logd$ samples? For any given point, half the functions would classify it correctly, and half wouldn't. For k points, approximately $\frac{d}{2^k}$ will classify all the points correctly. When we check the generalization error and try to minimize it, we will find it to be extremely large. This is the overfitting phenomenon.

1.1.8 Complexity versus Generalization

When choosing a model we have a tradeoff of hypothesis complexity versus observed error. More complex hypotheses have lower observed errors, but might have higher true errors.
A complex hypothesis that has many degrees of freedom would use all of them 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.

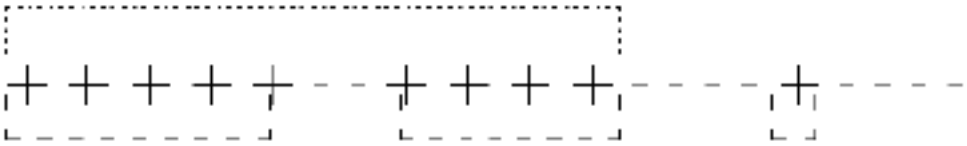

Figure 1.4: Two hypotheses for a given sample set. The dotted (upper) hypotheses is much better then the lower one, Though it's observed error on the sample data is higher

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*

$$\hat{\varepsilon} + \frac{|code\ length\ of\ h|}{m}$$

- *Structural Risk Minimization*

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

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

Using this formula we can calculate generalization errors.

1.1.9 Online Models

In each time t, we get a point x_t. We guess its classification y_t, and see the right classification $f(x_t)$.
Our goal is to minimize our number of errors, but this poses a problem - we can err each

and every time, since our opponent is the one that chooses the correct classification.
The solution to this problem is to assume a function $h^* \in H$ that always classifies correctly.
Our goal now is to approximate this function h^*, given $x_1, x_2, ..., x_T$.
The advantages of online models:

- We don't need a big sample to start classifying (though we'll probably make a lot of mistakes in the beginning).
- The hypothesis can be changed during the whole process.
- We make no assumptions about the distribution, and it can also be changed over time.

1.2 Course Structure

- 1. Basic models - PAC, Bayes
- 2. Online models - Perceptron, Regret, Boosting
- 3. Sampling complexity - VC-Dimension.
- 4. Efficient algorithms - Decision trees, Kernel + SVM, Fourier transform
- 5. Extra topics.

1.3 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".