E0 270 Machine Learning **Lecture 1 (Jan 8, 2013)**

Introduction. Binary Classification and Bayes Error.

Lecturer: Shivani Agarwal

Disclaimer: These notes are a *brief* summary of the topics covered in the lecture. They are not a substitute for the full lecture.

Outline

- Introduction and course overview
- Why study machine learning?
  - Examples of problems that can be solved using machine learning
  - What makes machine learning exciting
- Types of learning problems
  - Supervised learning: binary classification, multiclass classification, regression, ...
  - Unsupervised learning: clustering, density estimation, ...
  - Reinforcement learning
  - Several other useful variants (online learning, semi-supervised learning, active learning, ...)
- Binary classification and Bayes error

1 Introduction and Course Overview

See course information at:

`http://www.csa.iisc.ernet.in/~e0270/Jan-2013/`

2 Why Study Machine Learning?

Figure 1 shows some examples of problems where we might use machine learning techniques (indeed, machine learning techniques are already being used with success to solve these problems in practice). As can be seen, the problems are varied and come from a variety of different application domains.

The reasons that one might be interested in studying machine learning can be equally varied (see Figure 2). In this course, our reasons for studying machine learning are largely practical. Predictive models are needed in many areas of science, engineering, and business. With the explosion in data everywhere, ranging from astronomy, biology, and drug discovery to climate modeling, finance, and the Web, there is an increasing need for algorithms that can automatically "learn" such predictive models from data. In this course, we would like to understand how to design and analyze such algorithms.

That said, the foundations of machine learning are built on elegant mathematics from the fields of probability, statistics, computer science, and optimization, and it is through the right appreciation and understanding of these mathematical foundations that one can make lasting contributions to the development and analysis of new machine learning algorithms. To this end, while our focus will be on understanding the main tools and techniques in machine learning, we will emphasize a clear understanding of the underlying mathematical principles, with the hope that several of you may later go on to contribute new ideas to this area which also continues to be an active and exciting area of research with much potential for real impact.

Example 1: Email Spam Filter

- Would like to build an email filter that can predict whether a new message is spam or non-spam
- Have data containing previous examples of messages labeled as spam or non-spam
- Can we "learn" an email filter from this data?

Spam

Non-Spam

Spam

...

Example 2: Handwritten Digit Recognition

- Would like to build a model that can automatically recognize handwritten digits from images
- Have data containing examples of such images labeled with the correct digit (0,1,...,9)
- Can we "learn" an accurate recognition model from this data?

Example 3: Weather Forecasting

- Would like to build a forecasting model which, given a satellite image showing water vapor in a region, can predict the amount of rainfall in the coming week
- Have data containing examples of such images recorded in the past, together with the amount of rainfall observed in the following weeks
- Can we "learn" a forecasting model from this data?

22.1 mm

134.3 mm

47.9 mm

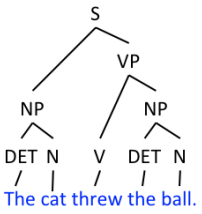
...

Example 4: Gene Expression Analysis

- Would like to identify genetic patterns in patients, such as groups of genes that have similar behavior, or groups of patients that have a similar form of genetic disease
- Have microarray expression data containing expression levels of thousands of genes in various patients
- Can we "learn" genetic patterns from this data?

Example 5: Natural Language Parsing

- Would like to build a natural language parser which, given an English sentence, can predict the correct parse tree for the sentence
- Have data containing examples of English sentences with their correct parse tree annotations
- Can we "learn" a natural language parser from this data?



Example 6: Movie Recommendations

- Would like to build a recommendation system that can recommend movies to users
- Have a (highly incomplete) movie rating matrix containing ratings (1-5 stars) provided by various users for movies they have watched
- Can we "learn" a recommendation system from this data?

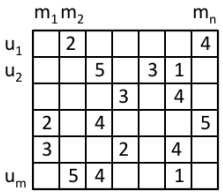


Figure 1: Examples of problems that can be solved using machine learning techniques.

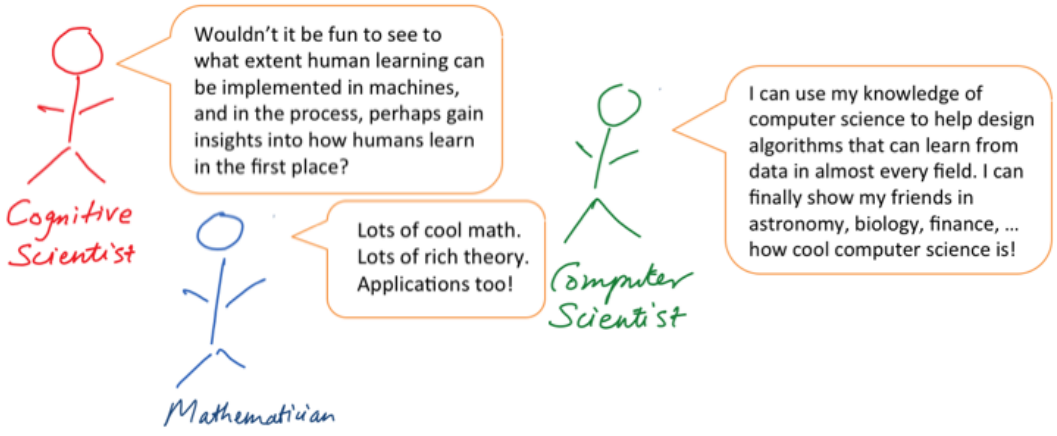


Figure 2: What makes machine learning exciting?

3 Types of Learning Problems

Here's the anatomy of a typical machine learning problem:[1]

Figure 3: Anatomy of a typical machine learning problem.

The input to the problem consists of training **data**; the output consists of a predictive (or explanatory or decision-making) **model**. In order to specify a learning problem, one needs to specify three critical components:

- the form of training data available as input;
- the form of model desired as output; and
- the **performance measure** or **evaluation criterion** that will be used to evaluate the model.

A candidate solution to the problem is then any **learning algorithm** that takes the specified form of data as input and produces the specified form of model as output. Algorithms that have stronger guarantees on the performance of the learned models in terms of the specified performance measure generally constitute better solutions than algorithms with weaker performance guarantees. In addition, there may be other considerations that need to be taken into account in designing a learning algorithm, e.g. time and space complexity of the algorithm, "interpretability" of the models produced, etc.

A prominent class of learning problems, and one that will be the focus of the first part of the course, is that of **supervised learning** problems. Here one is given examples of some objects together with associated labels, and the goal is to learn a model that can predict the labels of new objects; this includes for example the email spam filter, handwritten digit recognition, weather forecasting, and natural language parsing examples in Figure 1. In a typical supervised learning problem, there is an **instance space** $\mathcal{X}$ containing (descriptions of) instances or objects for which predictions are to be made, a **label space** $\mathcal{Y}$ from which labels are drawn, and a **prediction space** $\widehat{\mathcal{Y}}$ in which predictions are to be made (often $\widehat{\mathcal{Y}} = \mathcal{Y}$, but this is not always the case). The training data consists of a finite number of labeled examples $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, and the goal is to learn from these examples a model $h_S : \mathcal{X} \to \widehat{\mathcal{Y}}$ that given a new instance $x \in \mathcal{X}$, predicts $\widehat{y} = h_S(x) \in \widehat{\mathcal{Y}}$:

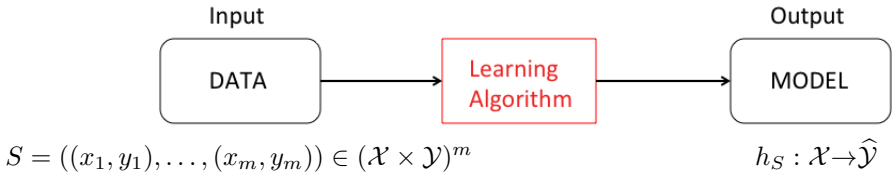


Figure 4: A typical supervised learning problem.

There are many types of supervised learning problems. For example, in a **binary classification** problem, there are two classes or labels, denoted without loss of generality by $\mathcal{Y} = \{\pm 1\} = \{-1, +1\}$, and the goal is to learn a model that can predict accurately the class (label) of new instances, $\widehat{\mathcal{Y}} = \mathcal{Y} = \{\pm 1\}$. The email spam filter problem in Figure 1 is an example of such a problem: here the two classes (labels) are spam and non-spam. The instance space $\mathcal{X}$ depends on the representation of the objects (in this case email messages): for example, if the email messages are represented as binary feature vectors denoting presence/absence of say some d words in a vocabulary, then the instance space would be $\mathcal{X} = \{0, 1\}^d$; if the messages are represented as feature vectors containing counts of the d words, then the instance space would be $\mathcal{X} = \mathbb{Z}_+^d$. In a

[1] As we'll see, not all machine learning problems follow this exact structure, but this is a good place to start.

multiclass classification problem, there are $r > 2$ classes (labels), say $\mathcal{Y} = \{1, \ldots, r\}$, and the goal again is to learn a model that predicts accurately labels of new instances, $\widehat{\mathcal{Y}} = \mathcal{Y} = \{1, \ldots, r\}$; the handwritten digit recognition problem in Figure 1 is an example of such a problem with $r = 10$ classes. In a **regression** problem, one has real-valued labels, $\mathcal{Y} \subseteq \mathbb{R}$, and the goal is to learn a model that predicts labels of new instances, $\widehat{\mathcal{Y}} = \mathcal{Y} \subseteq \mathbb{R}$; the weather forecasting problem in Figure 1 is an example of such a problem. In a **structured prediction** problem, the label space $\mathcal{Y}$ contains a potentially very large number of complex structures such as sequences, trees, or graphs, and the goal is generally to learn a model that predicts structures in the same space, $\widehat{\mathcal{Y}} = \mathcal{Y}$; the natural language parsing problem in Figure 1 is an example of such a problem. Performance in supervised learning problems is often (but not always) measured via a **loss function** of the form $\ell : \mathcal{Y} \times \widehat{\mathcal{Y}} \to \mathbb{R}_+$, where $\ell(y, \widehat{y})$ denotes the 'loss' incurred on predicting $\widehat{y}$ when the true label is y; we will see examples later.[2] Supervised learning techniques can also be extended to problems such as **collaborative filtering**, of which the movie recommendation problem in Figure 1 is an example.

In an **unsupervised learning** problem, there are no labels; one is simply given instances from some instance space $\mathcal{X}$, and the goal is to learn or discover some patterns or structure in the data; typically, one assumes the instances in the given training data $S = (x_1, \ldots, x_m) \in \mathcal{X}^m$ are drawn i.i.d. from some unknown probability distribution on $\mathcal{X}$, and one wishes to estimate some property of this distribution:

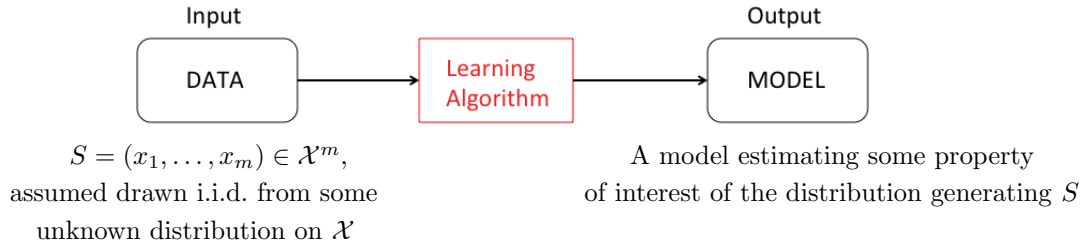


Figure 5: A typical unsupervised learning problem.

Unsupervised learning problems include for example **density estimation** problems, where one wants to estimate the probability distribution assumed to be generating the data S, and **clustering** problems, where one is interested in identifying natural 'groups' or 'clusters' in the distribution generating S. The gene expression analysis problem in Figure 1 falls in this category.

In a **reinforcement learning** problem, there is a set of states that an agent (learner) can be in, a set of actions that can be taken in each state, each of which leads to another state, and some form of 'reward' that the agent receives when it moves to a state; the goal is to learn a model, called a **policy**, that maps states to actions and determines which action the agent should take in each state, such that as an agent takes actions according to the model and moves from one state to another, the overall cumulative reward received is maximized. The difference from classical supervised or unsupervised learning problems is that the agent does not receive training data upfront, but rather learns the model while performing actions and observing their effects. Reinforcement learning problems arise for example in robotics, and in designing computer programs that can automatically learn to play games such as chess or backgammon.

There are also several useful variants of the above types of learning problems, such as **online learning** problems, where instances are received in a dynamic manner, the algorithm must predict a label for each instance as it arrives, and after each prediction, the true label of the instance is revealed and the algorithm can update its model; **semi-supervised learning** problems, where the goal is to learn a prediction model from a mix of labeled and unlabeled data; and **active learning** problems, where the algorithm is allowed to query labels of certain instances. We will see examples of all these later in the course.

As noted above, we will begin our study with supervised learning. We start with binary classification.

[2] Loss functions of the form $\ell : \mathcal{Y} \times \widehat{\mathcal{Y}} \to \mathbb{R}_+$ are known as **label-dependent** loss functions; for certain problems, other types of loss functions can be more appropriate.

4 Binary Classification and Bayes Error

Let $\mathcal{X}$ denote the instance space, and let the label space be $\mathcal{Y} = \{\pm 1\} = \{-1, +1\}$. We are given a **training sample** $S = ((x_1, y_1), \ldots, (x_m, y_m)) \in (\mathcal{X} \times \mathcal{Y})^m$, where each $x_i \in \mathcal{X}$ is an instance in $\mathcal{X}$ (such as a feature vector representing an email message in a spam classification problem, or a gene expression vector extracted from a patient's tissue sample in a cancer diagnosis problem), and $y_i \in \{\pm 1\}$ is a binary label representing which of two classes the instance x_i belongs to (such as spam/non-spam or cancer/non-cancer). Each pair (x_i, y_i) is referred to as a **(labeled) training example**. The goal is to learn from the training sample S a **classification model** or **classifier** $h_S : \mathcal{X} \to \{\pm 1\}$, which given a new instance (email message or tissue sample) $x \in \mathcal{X}$, can predict accurately its class label via $\widehat{y} = h_S(x)$.

What should count as a good classifier? In other words, how should we evaluate the quality of a proposed classifier $h : \mathcal{X} \to \{\pm 1\}$? We could count the fraction of instances in S whose labels are correctly predicted by h. But this would be a bad idea, since what we care about is not how well the classifier will perform on the given *training* examples, but how well it will perform on *new* examples in the future. How should we measure this?

Typically, S contains only a small number of the instances in $\mathcal{X}$. If $\mathcal{X}$ is finite, we could count the fraction of instances in $\mathcal{X}$ whose labels are predicted correctly by h, and use this as a measure of the quality of h. In general, however, $\mathcal{X}$ may be infinite. Also, some instances in $\mathcal{X}$ may appear more frequently than others (e.g. some email messages may be more likely to be seen than others), and in that case we would care more about predicting correctly on the more frequent instances. If we assume instances in future examples are generated according to some probability distribution μ on $\mathcal{X}$ and are labeled according to some true function $t : \mathcal{X} \to \{\pm 1\}$, then we could define the **accuracy of h with respect to μ, t** as the probability that an instance generated randomly from μ is classified correctly by h:

$$\mathrm{acc}_{\mu,t}[h] = \mathbf{P}_{x \sim \mu}\big(h(x) = t(x)\big)\,,$$

where the notation $\mathbf{P}_{x \sim \mu}(A)$ denotes the probability of an event A when the instance x is drawn randomly from μ. Equivalently, let us define the **0-1 loss function** $\ell_{0\text{-}1} : \{\pm 1\} \times \{\pm 1\} \to \{0, 1\}$ as follows:

$$\ell_{0\text{-}1}(y, \widehat{y}) = \mathbf{1}(\widehat{y} \neq y)\,,$$

where $\mathbf{1}(\cdot)$ is the indicator function whose value is 1 if its argument is true and 0 otherwise. Then we can define the **0-1 error of h w.r.t. μ, t** as

$$\mathrm{er}^{0\text{-}1}_{\mu,t}[h] = 1 - \mathrm{acc}_{\mu,t}[h] = \mathbf{P}_{x \sim \mu}\big(h(x) \neq t(x)\big) = \mathbf{E}_{x \sim \mu}\big[\mathbf{1}(h(x) \neq t(x))\big] = \mathbf{E}_{x \sim \mu}\big[\ell_{0\text{-}1}(t(x), h(x))\big]\,.$$

In general, it is possible that there is no 'true' label function t, in the sense that the same instance can sometimes appear with a $+1$ label and sometimes with a -1 label. This can happen for example if there is inherent noise or uncertainty in the underlying process, e.g. if the same gene expression vector can sometimes correspond to patients with a disease and sometimes to patients without the disease, or if the instances x do not contain all the information necessary to predict the outcome, e.g. if in addition to the gene expression measurements, one also needs some other information to make reliable predictions, in the absence of which both outcomes could potentially be seen. In this case we denote by $\eta(x) = \mathbf{P}(y = +1 | x)$ the conditional probability of seeing label $+1$ given instance x, and let D denote the resulting joint probability distribution over $\mathcal{X} \times \mathcal{Y}$. The **0-1 error of h w.r.t. D** is then defined as the probability that an example drawn randomly from D is misclassified by h:

$$\mathrm{er}^{0\text{-}1}_{D}[h] = \mathbf{P}_{(x,y) \sim D}\big(h(x) \neq y\big) = \mathbf{E}_{(x,y) \sim D}\big[\mathbf{1}(h(x) \neq y)\big] = \mathbf{E}_{(x,y) \sim D}\big[\ell_{0\text{-}1}(y, h(x))\big]\,.$$

Thus, given a training sample S, a good learning algorithm should produce a classifier h_S with small 0-1 error w.r.t. the underlying probability distribution D. How small can this error be? Clearly, $\mathrm{er}^{0\text{-}1}_{D}[h_S] \in [0, 1]$. Can we always aim for a classifier with zero error? If there is a 'true' label function $t : \mathcal{X} \to \{\pm 1\}$ such that every instance x appears only with label $y = t(x)$ (which means that for each x, $\eta(x)$ is either 0 or 1)[3], then

[3] Technically, we require this to hold with probability 1 over $x \sim \mu$.

clearly if $h_S = t$ we get zero error. In general, however, if the same instance x can appear with both $+1$ and -1 labels, then no classifier can achieve zero error. In particular, we have

$$\begin{aligned}
\mathrm{er}_D^{0\text{-}1}[h] &= \mathbf{E}_{(x,y)\sim D}\big[\mathbf{1}(h(x) \neq y)\big] \\
&= \mathbf{E}_{x\sim\mu}\big[\mathbf{E}_{y|x}\big[\mathbf{1}(h(x) \neq y)\big]\big] \\
&= \mathbf{E}_{x\sim\mu}\big[\mathbf{P}(y = +1|x) \cdot \mathbf{1}(h(x) \neq +1) + \mathbf{P}(y = -1|x) \cdot \mathbf{1}(h(x) \neq -1)\big] \\
&= \mathbf{E}_{x\sim\mu}\big[\eta(x) \cdot \mathbf{1}(h(x) = -1) + (1 - \eta(x)) \cdot \mathbf{1}(h(x) = +1)\big] .
\end{aligned}$$

For any x, a prediction $h(x) = -1$ contributes $\eta(x)$ to the above error, and a prediction $h(x) = 1$ contributes $(1 - \eta(x))$. The minimum achievable error, called the **Bayes error associated with** D, is therefore given by

$$\mathrm{er}_D^{0\text{-}1,*} \;=\; \inf_{h:\mathcal{X}\to\{\pm 1\}} \mathrm{er}_D^{0\text{-}1}[h] \;=\; \mathbf{E}_{x\sim\mu}\big[\min\big(\eta(x), (1 - \eta(x))\big)\big] .$$

Any classifier achieving the above error is called a **Bayes (optimal) classifier for** D; in particular, it follows from the above discussion that the classifier $h^* : \mathcal{X}\to\{\pm 1\}$ defined as

$$h^*(x) \;=\; \mathrm{sign}\big(\eta(x) - \tfrac{1}{2}\big) \;=\; \begin{cases} +1 & \text{if } \eta(x) > \frac{1}{2} \\ -1 & \text{otherwise} \end{cases}$$

is a Bayes optimal classifier for D.

Thus, if we know the underlying probability distribution D from which future examples will be generated, then all we need to do is use a Bayes classifier for D as above, since this has the least possible error w.r.t. D. In practice, however, we generally do not know D; all we are given is the training sample S. It is then usual to assume that the training examples in S are also generated independently from the same distribution D, and do one of the following:

- Assume a parametric form for D, such that given the parameters, one can compute a Bayes optimal classifier; the parameters are then estimated using the training sample S, which constitutes an i.i.d. sample from D, and a Bayes optimal classifier w.r.t. the estimated distribution is used;
- Assume a parametric form for the classification model, i.e. some parametric function class $\mathcal{H} \subseteq \{\pm 1\}^{\mathcal{X}}$, and use S to find a good classifier h_S within $\mathcal{H}$;
- Use 'non-parametric' methods to learn a classifier h_S from S.

In the next few lectures, we will discuss a variety of learning algorithms in each of the above categories. At the end of the course, we will see that even without knowledge of D, many of these algorithms can be made **(universally) statistically consistent** w.r.t. the 0-1 error, in the sense that whatever the distribution D might be, as the number m of training examples in S (drawn i.i.d. from D) goes to infinity, the 0-1 error of the learned classifier is guaranteed to converge in probability to the Bayes error for D.