CS142: Machine Learning Spring 2017

Lecture 1

Instructor: Pedro Felzenszwalb *Scribes: Dan Xiang, Tyler Dae Devlin*

Introduction

Some logistical details

- Prerequisites: Linear algebra (MATH 0520), probability & statistics (APMA 1650), and introductory computer science (CSCI 0150, 0160).
- This course focuses on the mathematical foundations of machine learning. An alternative, more practically oriented course is CS1951A: Data Science.
- This course's website is `http://cs.brown.edu/~pff/engn2520/`.
- The class Piazza page is `www.piazza.com/brown/spring2017/cs142`.

Supervised learning

The goal of *supervised learning* is to estimate a function $f : X \to Y$ where X is the "feature space" and Y is the "label space". The function f is estimated using a set of labeled training data

$$T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

with $\{x_i\} \subset X$ and $\{y_i\} \subset Y$, where each $y_i = f(x_i)$ is the true label for the corresponding x_i.

Contrast this to *unsupervised learning*, in which examples don't come with a set of know labels, i.e. we don't specify an output space. Instead of trying to estimate a target function f, we just wish to analyze the structure of the data $\{x_1, \ldots, x_n\}$.

Example. Let the input space be $X = \mathbb{R}^2$, let the output space be $Y = \{\text{Bass}, \text{ Salmon}\}$, and suppose each example $(x_1, x_2) \in X$ represents the length and weight of a fish. Given training data, we can estimate $f : X \to Y$, the function that determines whether a fish is a bass or salmon based on its length and weight. We can then use this estimate of f to classify new fish after weighing and measuring them.

Suppose we have some training data T, consisting of the lengths and weights of a bunch of fish along with their true species label. Define the classifier

$$g(x_1, x_2) = \begin{cases} \text{Salmon} & x_1 \geq t \\ \text{Bass} & x_1 < t \end{cases}.$$

The classifier g is an estimate of the target function f. Note that this is probably a poor estimate, since g classifies fish based only on their length, even though the training data also includes information about weight.

Nevertheless, to fully specify the classifier g we need to select the threshold t. Intuitively, we should select t to minimize the number of mistakes the classifier makes on the examples in the training data set, i.e. the **training error**. What we are really interested in, however, is the number of mistakes our function would make on a new set of unseen data, i.e. the **test error**. In general, it is easy to devise a classifier that has low (or even zero) training error; this is called *memorizing*. It is much harder to construct a classifier that generalizes to new data; this is called *learning*. $\square$

Theorem: If the number of samples n is large enough, then the training error $\approx$ test error with high probability. (We will make this precise later in the course.)

Leaving the fish classification example aside for the moment, let us examine a different kind of classifier. Suppose we have a training set $T = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. Define the classifier

$$g(x) = \begin{cases} y_j & x = x_j \text{ for some } j \in \{1, \ldots, n\} \\ 0 & x \neq x_j \text{ for all } j \in \{1, \ldots, n\} \end{cases}.$$

In words, this classifier checks if the feature x was observed in the training data. If it was, then it assigns the corresponding observed y_j label. Otherwise, $g = 0$.

Example. Classification is a very common task. A classic example is spam detection. In this case,

$$X = \text{set of all possible emails},$$
$$Y = \{\text{spam}, \ \text{not spam}\}.$$

We assume there is some true function f that associates to each email a label of either "spam" or "not spam". Our goal is to use labeled training examples to construct an estimate g of f so that we can classify new emails and place them in the appropriate folders in your mail client.

$\square$

Formalizing the problem

Given a feature space X and a label space Y, there is some distribution/density $p(x, y)$ over $X \times Y$ (the Cartesian product of X and Y). We define a loss function $L : Y \times Y \to \mathbb{R}$ by

$$L(y, \hat{y}) = \text{ cost of predicting } \hat{y} \text{ if the true label is } y.$$

A quantity of interest is the expected loss of predictions using our learend classifier g, denoted

$$E[L(y, g(x))] \tag{1}$$

where y is the true label, $g(x)$ is our predicted label, and $(x, y) \sim p(x, y)$, .

We can define a particular loss function

$$L(y, \hat{y}) = \begin{cases} 0 & y = \hat{y} \\ 1 & y \neq \hat{y} \end{cases}.$$

Then equation (1) is the probability of error (since the expectation of an indicator of an event is precisely the probability of that event).

We typically assume that a training set T is a collection of iid samples drawn according the joint distribution $p(x, y)$. Then we have that expression (1) is roughly equal to

$$\frac{1}{n} \sum_{i=1}^{n} L(y_i, g(x_i)),$$

i.e. the sample mean of L, also known as the empirical loss.

Given the underlying distribution $p(x, y)$, we compute using the definition of expected value

$$E[L(y, g(x))] = \int_X \sum_{y \in Y} p(x, y) L(y, g(x))\, dx$$

For each x, we define the classifier

$$g(x) = \underset{\hat{y} \in Y}{\operatorname{argmin}} \sum_{y \in Y} p(x, y) L(y, \hat{y}).$$

In this case we can use the training data T to estimate the joint distribution $p(x, y)$. We can write

$$p(x, y) = p(y) p(x \mid y).$$

We can estimate $p(y)$ by observing how many instances of each class $y \in Y$ occur in T. For the $p(x \mid y)$ term, we can estimate $p(x \mid \text{Salmon})$ and $p(x \mid \text{Bass})$ as Gaussians (for example) using parameters inferred from the training data.