Cornell Machine Learning Notes

Bryan Gass

November 24 2020

Contents

1 Supervised Learning

high level overview of supervised machine learning: attempts to make predictions from data

1.1 Setup

We are given a dataset **D** within data set we have data points and their labels, where x_i is D-dimensional vector. x_i is considered to be an input instance/attribute/feature/covariate. x_i also can be something much more complex like an image, a sentence or email, or molecular shape. y_i is the label associated with that feature vector/input. There are also **n** data points within a dataset, **D**.

It should be noted the difference between x and x_i as well as y and y_i. x_i can be thought of the **feature vector** that we are looking to train our model on, while generally speaking x is the feature vector that behaves as a testing input, thus why we refer to it as a **test input**. This idea carries over to y and y_i. y_i is the label that we use to check the validity of our feature vector x_i. y is the output or the **response variable** that is associated with the test input.

$$D = (x_1, y_1), ..., (x_n, y_n) \tag{1}$$

- $\boldsymbol{R}^d$ is the d-dimensional feature space
- x_i is the input vector of the i^{th} sample
- y_i is the label of the i^{th} sample
- C is the label space

The arbitrary data point x_i and the label y_i is assumed to be under the normal distribution P. It should be noted we do not have access to this normal distribution, P, ever. It is a hypothetical normal distribution that contains the theoretical total amount of data of the dataset. Where our dataset **D** is within P.

$$(x_i, y_i) \sim P \tag{2}$$

The Idea of Supervised Learning: to take our data set that contains the data points, x_i, and our labels ,y_i, to "learn" a function that can goes from x_i to y_i. This also can be thought of as trying to "learn" a model such that it can adequately map inputs x to outputs y, given by a input-output pair, our training set. Denoted as:

$$D = (x_i, y_i)_{i=1}^{N} \tag{3}$$

Where D is called the training set and N is the number of training examples

Term 1: **i.i.d.** - Independent And Identically Distributed Random variables

1.2 Supervised Learning:

Most methods will assume that y_i, the label, is either a categorical or nominal variable. given $y_i \in 1, ..., C$ When y_i is categorical (male or female) then it is considered a **classification problem**. When y_i is a real valued scalar (like house prices) is considered to be a **regression problem**. Another variant of this is when y_i has some type of natural ordering to it, such as grade values from A to F, this is considered to be **ordinal regression**.

- **Binary Classification:** if $C = 2$ then this is considered to be a binary classification problem

  $$y \in 0, 1 \text{ or } y \in -1, +1 \tag{4}$$

  Predicting whether or not a given input belongs to one of two classes, typically in terms of true or false, or positive or negative. (e.g., face detection).

  1. Example 1: Given a classifier that determines if something is spam or not we could view x as the email and y as a label that is either "spam" or "not spam"
  2. Example 2: Given a classifier that determines if the stock market is rising or falling we can view the x as the stock and the y as the label that is either "up" or "down"

- **Regression:**

  $$Y \in \mathbb{R} \tag{5}$$

  Predicting a real number. (e.g., predicting gas prices).

- **Multi-class Classification:**

  $$y \in 1, ..., C \text{ Where C is a discrete number.} \tag{6}$$

  Predicting which of K classes an input belongs to. (e.g., face recognition).

- **Multi-label Classification** When there are many class labels that are not mutually exclusive to one another (tall but not strong). Generally speaking this is comparing many non-mutually exclusive binary classifications to one another, but is not always the case.

1.3 Unsupervised Learning:

Within unsupervised learning we are only given the feature vectors x_i, without the label y_i, where we are tasked to find "interesting patterns". This is often called **Knowledge Discovery**. This is denoted by:

$$D = (x_i)_{i=1}^{N} \tag{7}$$

1.4 Training vs. Testing Data

- **Training:** A learner L is given a set of training data (x1,y1),. . . ,(xn,yn) and allowed to perform computations on the training data to learn an output function h(x).
- **Testing:** L is asked to generate predictions $h(x_1)$, ..., $h(x_m)$ for a set of testing data. We can then compute an error metric or loss function to determine quantitatively if the learner correctly predicted the true outputs for the test examples. A common metric for classification tasks is the 0–1 loss, which is just the proportion of incorrect guesses

1.5 Hypothesis Functions

A list of arbitrary functions that we the scientists can pick from that we believe will best be able to fit the data. That is, the function that is best at taking and input and finding the associated label within a test setting

we can describe the function we choose to be h within the arbitrarily large list of H functions which are known to be **Hypothesis Classes**.

1.6 Loss Functions:

1. **0/1 Loss Function**: We run this particular h over our dataset, D.

   $$L_{0/1}(h) = \frac{1}{n}\sum_{i=1}^{n} \delta_{h(x_i)\neq y} \tag{8}$$

   where

   $$\delta_{h(x_i)\neq y} = \begin{cases} 1, & \text{if } h(x_i) \neq y \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

   For every single example it suffers a loss of 1 if it is mispredicted, and 0 otherwise. The normalized zero-one loss returns the fraction of misclassified training samples, also often referred to as the training error.

   The loss function returns the error rate on this data set D.

2. **Squared Loss:** Typically used within regression, the square loss can be thought of as prediction minus the actual value. However there are trade offs made by using the square loss function. The loss suffered is always

non-negative. Additionally, due to the square loss being quadratic that means outliers that made be off by a lot may make it seem like this function is doing worse than, lets say, a lot of points off by a little.

$$h(x) = E_{P(y|x)}[y] \tag{10}$$

$$L_{sq}(h) = \frac{1}{n}\sum_{i=1}^{n}(h(x_i) - y_i)^2 \tag{11}$$

3. **Absolute Loss:** to remedy the effect outliers within our potential dataset, D, we can you the absolute loss function that will not square the result but rather take the absolute value. This means that the suffered loss grows linearly

$$h(x) = \mathbf{MEDIAN}_{P(y|x)}[y] \tag{12}$$

$$L_{abs}(h) = \frac{1}{n}\sum_{i=1}^{n} \mid h(x_i) - y_i \mid \tag{13}$$

2 k-Nearest Neighbors

Remarks to keep in mind:

- choose an odd k value for a 2 class problem
- k must not be a multiple of the number of classes
- the main drawback of kNN is the complexity in searching the nearest neighbors for each sample