Machine Learning: A Probabilistic Perspective

immediate

January 4, 2019

1 Introduction

1.1 Types of machine learning

Predictive or Supervised Learning The goal is to learn a mapping from inputs to outputs given a labeled set of input-output pairs.

Descriptive or Unsupervised Learning The goal is to find interesting patterns in a given dataset.

1.2 Supervised Learning

Classification Learning a mapping from inputs x to outputs y where $y \in \{1, ..., C\}$, with C being the number of classes. Classification can be formalized as **function approximation**. Assume $y = f(x)$ for some unknown function f. We estimate f given a labeled training set, and make predictions from the estimated function, which we denote $\hat{f}(x)$.

Binary Classification Where the number of classes is 2.

Multiclass Classification Where the number of classes is greater than 2.

Multi-label classification Where each instance may belong to multiple classes.

Generalization Making predictions on inputs not in the training set.

Probabilistic Predictions Given a model that outputs a probability for each class, we can make our "best guess" for the class of the instance by returning the most probable class label; the mode of the distribution. This is called the **MAP Prediction** or **Maximum A Posteriori**, meaning that the class has the highest likelihood given the instance.

$$\hat{y} = \hat{f}(x) = argmax_c \, p(y = c|x, D) \tag{1}$$

where D is the training dataset.

Real World Applications of Classification

- Document Classification
- Spam filtering
- Image Classification
- Handwriting Recognition
- Face detection and Recognition

Regression Like classification except that the response variable you are trying to predict is continuous.

1.3 Unsupervised Learning

The goal is to discover *structure* in the data; this is sometimes called *knowledge discovery*. Can be formalized as **density estimation**, meaning building models that return a probability density for each input: $p(x_i|\theta)$. **Supervised and unsupervised learning can be distinguished as conditional and unconditional density estimation, respectively.**

1.3.1 Clustering

The problem of separating data into groups. Let K denote the number of clusters. The first goal is to estimate a probability distribution over the number of clusters, $p(K|D)$. The second goal is to estimate which cluster each point belongs to. The cluster each point belongs to would be known as a *latent* or *hidden* variable, because it isn't observed directly in the data.

Discovering Latent Factors It's often useful to reduce the dimensionality of data to a lower dimensional subspace that caputres the "essence" of the data. This is called **dimensionality reduction**.

The motivation is that the raw data may be high dimensional but there may be only a small number of degrees of variability, corresponding to latent factors.

Discovering Graph Structure Sometimes we measure a set of correlated variables, and we would like to discover which ones are most correlated with others. This can be represented by a graph G, in which nodes represent variables, and edges represent direct dependence between variables

Matrix Completion Sometimes we have missing information, and inferring plausible values for missing values from existing values is called **imputation**. This is sometimes called *matrix completion*. An example application is *image inpainting*, in which holes in an image are filled in with realistic textures and shapes.

Inferring values in an extremely large, extremely sparse matrix is sometimes called *collaborative filtering*.

1.4 Some basic concepts in machine learning

1.4.1 Parametric vs Non-parametric models

Parametric models have a fixed number of parameters. Non-parametric models have a number of parameters that scales with the size of the dataset.

Parametric models have the advantage of being faster to use, but making stronger assumptions about the nature of the data generating distribution.

1.4.2 A simple non-parametric classifier: K-nearest neighbors

Checks the K points which are nearest to the query instance x, counts how many members of each class there are in this set, and returns that empirical fraction as the estimate. This is an example of **memory-based learning** or **instance-based learning**.

A KNN classifier with $K = 1$ induces a **Voronoi Tesselation** of the feature space, in which each point x_i is associated with a partition of the space such that all points in the partition are closer to that x_i than any other point in the training set.

It can be shown that the KNN classifier approaches a factor of 2 of the best possible classifier performance as the number of points in the training set approaches infinity.

1.4.3 The curse of dimensionality

As the number of dimensions of the feature space increases, the average instance density must decrease. Similarly, the average distance of the query instance to training instances must increase. This means that there are fewer points nearby to learn from when making an estimate on the query instance, and that it's harder to distinguish signal from noise, because irrelevant features "distract" learning algorithms. Information "local" to the query instance no longer is very local, because if you make your estimate from a fixed fraction of the data, the distance from the query instance needed to obtain that data increases dramatically. Even with just 3 features, if you want to estimate from 10% of the training data, you need to pull data from a hypercube whose side length is half the width of the entire feature space. Hence the importance of dimensionality reduction.

1.4.4 Parametric models for classification and regression

The main way to combat the curse of dimensionality is to make some assumptions of the nature of the data generating distribution. These assumptions are known as **inductive bias** and are often embodied in the form of a **parametric model**.

Linear regression Asserts that the output is a linear function of the inputs:

$$y(x) = w^T x + \epsilon, \tag{2}$$

where w is a weight vector, x is the input vector, and $epsilon$ represents the *residual error* between the model's predictions and the true output.

It is often assumed that the residual error is normally distributed. This is denoted $\epsilon \sim N(\mu, \sigma^2)$.

To make it clear that a model induces a conditional probability density, we may write:

$$p(y|x, \theta) = N(y|\mu(x), \sigma^2(x)), \tag{3}$$

In general, μ can be any function of x - linear regression can model non-linear relationships by making it a nonlinear function. This is called **basis function expansion**. In standard linear regression $\mu(x) = w^T x$.

Many machine learning methods are actually just different ways of inducing the basis function from the training data.

Extending the input vector to simplify the use of a bias term A common notational practice is to extend the input vector so that it includes a term of constant value 1, so that $x = (1, x)$. This allows the bias term (like a y-intercept for a linear function) to be included in the weight vector and simplifies many formulas.