COMP 551: Applied Machine Learning Notes

Brendan Kellam

2020
September

1 Preamble

This document will act as a running set of notes for COMP 551, Applied Machine Learning [2]. My objective with this document is to provide a summarized version of the course.

2 Introduction to Machine Learning

What exactly **is** ”machine learning”? I like Kevin P. Murphy’s definition from his book ”Machine Learning: A Probabilistic Perspective”:

> ”In particular, we define machine learning as a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (such as planning how to collect more data!)” [1]

To solve such problems, we can utilize the tools of probability theory.

A summarized review of probability theory would be useful.

3 Types of machine learning

In the following section, we will review the main ”types” of machine learning and go over a brief introduction into the different variations for each.

3.1 Supervised / predictive learning

Goal: learn a mapping from inputs $\mathbf{x}$ to outputs y
Given:

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}.$$

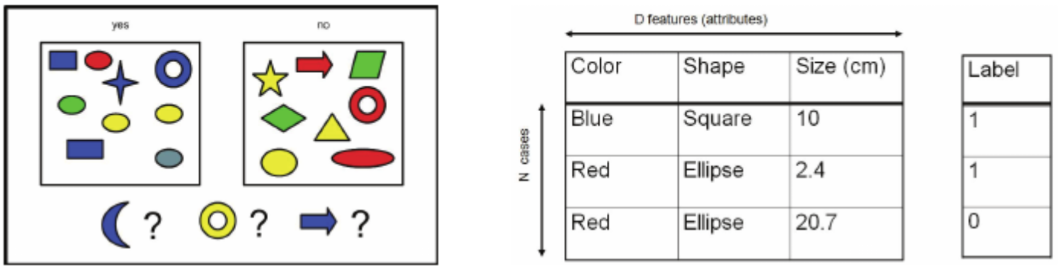
Where:

- $\mathcal{D}$ – training set
- N – number of training examples
- $\mathbf{x}_i$ – input (e.g. a D-dimensional vector). Each element $\in \mathbf{x}$ is known as a **feature/attribute/covariant**. $\mathbf{x}_i$ could be a complex structured object, like a image, sentence, etc.
- y_i – output/response variable. y_i is typically either a **categorical/nominal** variable from a finite set $y_i \in \{1, \ldots, C\}$ (e.g. $y_i \in \{male, female\}$) **or** a **real-valued scalar** $y_i \in \mathbb{R}$ (e.g. income level).

Note: If y_i is categorical $\implies$ the problem is known as **classification/pattern recognition**. Otherwise, if y_i is real-valued $\implies$ the problem is **regression**.

3.1.1 Classification

We can formalize the problem of classification by viewing it via the lense of **function approximation** [1]. We assume $y = f(\mathbf{x})$ for some unknown function f. The goal of **learning** is to estimate this function f given a labeled training set, and then to make predictions using $\hat{y} = \hat{f}(\mathbf{x})$ (Where the hat denotes a estimate.) We can then use our estimate $\hat{f}$ to make predictions on inputs that do not exist within our training set (i.e. some $\mathbf{x}$ s.t. $\mathbf{x} \notin \mathcal{D}$).
Example:
Figure 1 depicts a simple classification example where we are tying to classify objects with three features ({color, shape, size} $\in \mathbf{x_i}$) into two catagories ({yes, no} $\in y_i$).



Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

(a)

(b)

Figure 1: (a) Labeled training examples along with 3 unlabeled test cases. (b) NxD design matrix representing our training data. Each row is a feature vector $\mathbf{x}_i$. Last column is the label, $y_i \in \{0, 1\}$ [1]

In the example, predicting the label for the moon is quite clear (it will likely be assigned a "yes" label since all training examples with color blue exist within this category). A classification for the yellow circle is less certain: there are training examples with color yellow and shape circle in both catagories. This notion of ambiguity in our estimations motivates the need for a probability metric.

The probability distribution over possible labels, given input vector $\mathbf{x}$ and training set $\mathcal{D}$, is given by $p(y|\mathbf{x}, \mathcal{D})$. This represents a vector of length C (e.g. given $\mathbf{x}_i$ = [Blue, Square, 10], we may expect $p(y|\mathbf{x}_i, \mathcal{D})$ = [0.2, 0.8], implying we are 80% sure $\mathbf{x}_i$ should be labeled as 1). We can compute our "best guess" as to the "true label" by using:

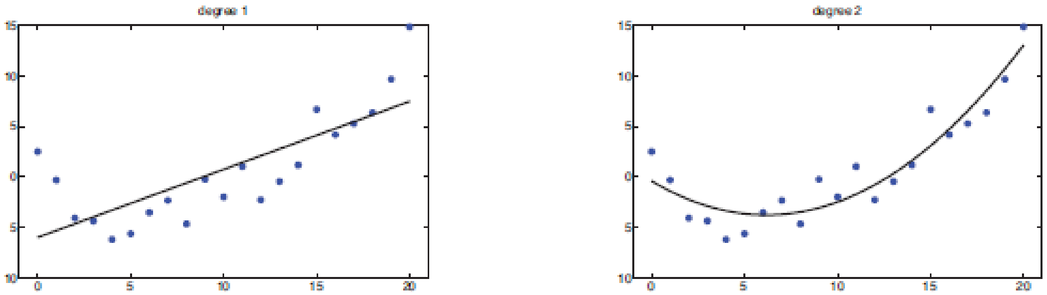
$$\hat{y} = \hat{f}(\mathbf{x}) = \operatorname*{argmax}_{c=1}^{C} p(y = c|\mathbf{x}, \mathcal{D}) \tag{1}$$

(1) corresponds to the most probable class label and is also known as the **MAP** estimate (Maximum A Posteriori). In the case of the yellow circle from Figure 1, we could expect $p(\hat{y}|\mathbf{x}, \mathcal{D})$ being far from 1.0 (i.e. we are not confident of our classification).

NOTE: we are implicitly conditioning our classification probabilities on the chosen model. When needed, we can explicitly denote a conditional on a model by writing $p(y|\mathbf{x}, \mathcal{D}, M)$ where M is our chosen model.

3.1.2 Regression

Regression is the same as classification, except the response variable/output is continuous.



(a) (b)

Figure 2: (a) Labeled training examples along with 3 unlabeled test cases. (b) NxD design matrix representing our training data. Each row is a feature vector $\mathbf{x}_i$. Last column is the label, $y_i \in \{0, 1\}$ [1]

3.2 Unsupervised / descriptive learning

Goal: learn a mapping from inputs $\mathbf{x}$ to outputs y
Given:

$$\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{N}.$$

3.3 Reinforcement learning

Goal: Learn how to act/behave when given occasional reward or punishment signals.

4 KNN

TODO

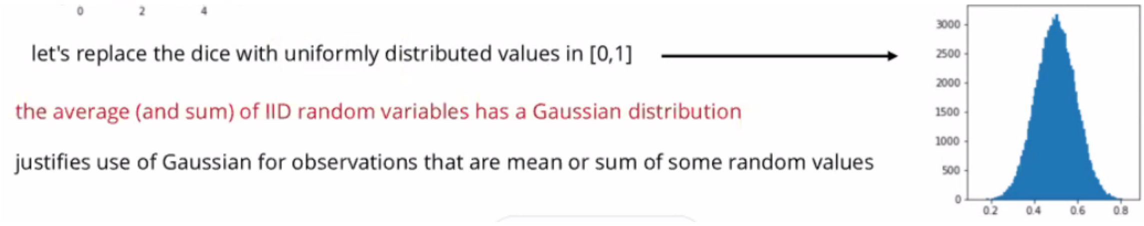
5 Multivariate Gaussian

Gaussian probability density function (PDF):

$$\mathcal{N}(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Two reasons why Gaussian is an important dist.

1. Maximum entropy dist. with a fixed variance
2. Central limit theorem:



let's replace the dice with uniformly distributed values in [0,1] →

the average (and sum) of IID random variables has a Gaussian distribution

justifies use of Gaussian for observations that are mean or sum of some random values

Multivariate Gaussian:

$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi\boldsymbol{\Sigma})}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$$

References

[1] Kevin P. Murphy. *Machine Learning : A Probabilistic Perspective.* MIT Press, 2012. ISBN: 0262018020.

[2] Siamak Ravanbakhsh. *Applied Machine Learning. COMP 551 - Fall 2020.* URL: `https://www.siamak.page/teachings/comp551f20/comp551f20/`.