CHAPTER 13

Machine Learning Concepts

13.1 Styles of Learning

Broadly speaking the main two subfields of machine learning are *supervised learning* and *unsupervised learning*. In supervised learning the focus is on accurate prediction, whereas in unsupervised learning the aim is to find accurate compact descriptions of the data.

Particularly in supervised learning, one is interested in methods that perform well on previously unseen data. That is, the method 'generalises' to unseen data. In this sense, one distinguishes between data that is used to train a model, and data that is used to test the performance of the trained model, see fig(13.1).

13.1.1 Supervised learning

Consider a database of face images, each represented by a vector[1] $\mathbf{x}$. Along with each image $\mathbf{x}$ is an output class $y \in \{\mathsf{male}, \mathsf{female}\}$ that states if the image is of a male or female. A database of 10000 such image-class pairs is available, $\mathcal{D} = \{(\mathbf{x}^n, y^n), n = 1, \ldots, 10000\}$. The task is to make an accurate predictor $y(\mathbf{x}^*)$ of the sex of a novel image $\mathbf{x}^*$. This is an example application that would be hard to program in a traditional 'programming' manner since formally specifying how male faces differ from female faces is difficult. An alternative is to give examples faces and their gender labels and let a machine automatically 'learn' a rule to differentiate male from female faces.

Definition 88 (Supervised Learning)**.** Given a set of data $\mathcal{D} = \{(x^n, y^n), n = 1, \ldots, N\}$ the task is to 'learn' the relationship between the input x and output y such that, when given a new input x^* the predicted output y^* is accurate. To specify explicitly what accuracy means one defines a loss function $L(y^{pred}, y^{true})$ or, conversely, a utility function $U = -L$.

In supervised learning our interest is describing y conditioned on knowing x. From a probabilistic modelling perspective, we are therefore concerned primarily with the conditional distribution $p(y|x, \mathcal{D})$. The term 'supervised' indicates that there is a 'supervisor' specifying the output y for each input x in the available data $\mathcal{D}$. The output is also called a 'label', particularly when discussing classification.

Predicting tomorrow's stock price $y(T+1)$ based on past observations $y(1), \ldots, y(T)$ is a form of supervised learning. We have a collection of times and prices $\mathcal{D} = \{(t, y(t)), t = 1, \ldots, T\}$ where time t is the 'input' and the price $y(t)$ is the output.

[1]For an $m \times n$ face image with elements F_{mn} we can form a vector by stacking the entries of the matrix. In MATLAB one may achieve this using `x=F(:)`.

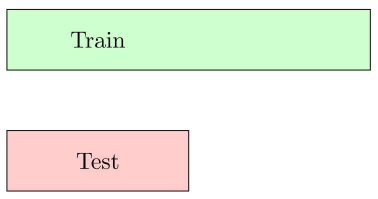


Figure 13.1: In training and evaluating a model, conceptually there are two sources of data. The parameters of the model are set on the basis of the train data only. If the test data is generated from the same underlying process that generated the train data, an unbiased estimate of the generalisation performance can be obtained by measuring the test data performance of the trained model. Importantly, the test performance should not be used to adjust the model parameters since we would then no longer have an independent measure of the performance of the model.

Example 57. A father decides to teach his young son what a sports car is. Finding it difficult to explain in words, he decides to give some examples. They stand on a motorway bridge and, as each car passes underneath, the father cries out 'that's a sports car!' when a sports car passes by. After ten minutes, the father asks his son if he's understood what a sports car is. The son says, 'sure, it's easy'. An old red VW Beetle passes by, and the son shouts – 'that's a sports car!'. Dejected, the father asks – 'why do you say that?'. 'Because all sports cars are red!', replies the son.

This is an example scenario for supervised learning. Here the father plays the role of the supervisor, and his son is the 'student' (or 'learner'). It's indicative of the kinds of problems encountered in machine learning in that it is not really clear anyway what a sports car is – if we knew that, then we wouldn't need to go through the process of learning. This example also highlights the issue that there is a difference between performing well on training data and performing well on novel test data. The main interest in supervised learning is to discover an underlying rule that will generalise well, leading to accurate prediction on new inputs.

For an input x, if the output is one of a discrete number of possible 'classes', this is called a *classification problem*. In classification problems we will generally use c for the output.

For an input x, if the output is continuous, this is called a *regression problem*. For example, based on historical information of demand for sun-cream in your supermarket, you are asked to predict the demand for the next month. In some cases it is possible to discretise a continuous output and then consider a corresponding classification problem. However, in other cases it is impractical or unnatural to do this; for example if the output y is a high dimensional continuous valued vector, or if the ordering of states of the variable is meaningful.

13.1.2 Unsupervised learning

Definition 89 (Unsupervised learning)**.** Given a set of data $\mathcal{D} = \{x^n, n = 1, \ldots, N\}$ in unsupervised learning we aim to to 'learn' a plausible compact description of the data. An objective is used to quantify the accuracy of the description.

In unsupervised learning there is no special 'prediction' variable. From a probabilistic perspective we are interested in modelling the distribution $p(x)$. The likelihood of the data under the i.i.d. assumption, for example, would be one objective measure of the accuracy of the description.

Example 58. A supermarket chain wishes to discover how many different basic consumer buying behaviours there are based on a large database of supermarket checkout data. Items brought by a customer on a visit to a checkout are represented by a (very sparse) 10,000 dimensional vector $\mathbf{x}$ which contains a 1 in the i^{th} element if the customer bought product i and 0 otherwise. Based on 10 million such checkout vectors from stores across the country, $\mathcal{D} = \left\{\mathbf{x}^n, n = 1, \ldots, 10^7\right\}$ the supermarket chain wishes to discover patterns of buying behaviour.

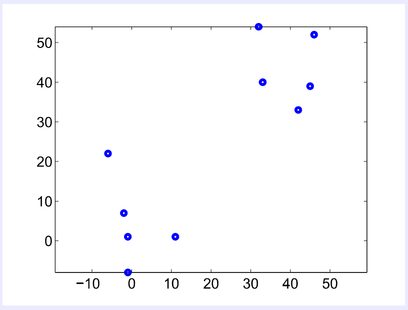
In the table each column represents the buying patterns of a customer (7 customer records and just the first 6 of the 10,000 products are shown). A 1 indicates that the customer bought that item. We wish to find common patterns in the data, such as if someone buys coffee they are also likely to buy milk.

coffee	1	0	0	1	0	0	0	· ·
tea	0	0	1	0	0	0	0	· ·
milk	1	0	1	1	0	1	1	· ·
beer	0	0	0	1	1	0	1	· ·
diapers	0	0	1	0	1	0	1	· ·
aspirin	0	1	0	0	1	0	1	· ·

Example 59 (Clustering)**.**

The table on the right represents a collection of unlabelled two-dimensional points. We can visualise this data by plotting it in 2 dimensions.

x_1	-2	-6	-1	11	-1	46	33	42	32	45
x_2	7	22	1	1	-8	52	40	33	54	39



By simply eye-balling the data, we can see that there are two apparent clusters here, one centred around (0,5) and the other around (35,45). A reasonable model to describe this data might therefore be to describe it as two clusters, centred at (0,0) and (35,35), each with a standard deviation of around 10.

13.1.3 Anomaly detection

A baby processes a mass of initially confusing sensory data. After a while the baby begins to understand her environment in the sense that novel sensory data from the same environment is familiar or expected. When a strange face presents itself, the baby recognises that this is not familiar and may be upset. The baby has learned a representation of the familiar and can distinguish the expected from the unexpected; this is an example of unsupervised learning. Models that can detect irregular events are used in *plant monitoring* and require a model of normality which will in most cases be based on unlabelled data.

13.1.4 Online (sequential) learning

In the above situations, we assumed that the data $\mathcal{D}$ was given beforehand. In *online learning* data arrives sequentially and we want to continually update our model as new data becomes available. Online learning may occur in either a supervised or unsupervised context.

13.1.5 Interacting with the environment

In many real-world situations, an agent is able to interact in some manner with its environment.

Query (Active) Learning Here the agent has the ability to request data from the environment. For example, a predictor might recognise that it is less confidently able to predict in certain regions of the space x and therefore requests more training data in this region. Active Learning can also be considered in an unsupervised context in which the agent might request information in regions where $p(x)$ looks uninformative or 'flat'.

Reinforcement Learning One might term this also 'survival learning'. One has in mind scenarios such as encountered in real-life where an organism needs to learn the best actions to take in its environment in order to survive as long as possible. In each situation in which the agent finds itself it needs to take an action. Some actions may eventually be beneficial (lead to food for example), whilst others may be disastrous (lead to being eaten for example). Based on accumulated experience, the agent needs to learn which action to take in a given situation in order to obtain a desired long term goal. Essentially actions that lead to long term rewards need to reinforced. Reinforcement learning has connections with control theory, Markov decision processes and game theory. Whilst we discussed MDPs and briefly mentioned how an environment can be learned based on delayed rewards in section(7.8.3), we will not discuss this topic further in this book.

13.1.6 Semi-supervised learning

In machine learning, a common scenario is to have a small amount of labelled and a large amount of unlabelled data. For example, it may be that we have access to many images of faces; however, only a small number of them may have been labelled as instances of known faces. In semi-supervised learning, one tries to use the unlabelled data to make a better classifier than that based on the labelled data alone.

13.2 Supervised Learning

Supervised and unsupervised learning are mature fields with a wide range of practical tools and associated theoretical analyses. Our aim here is to give a brief introduction to the issues and 'philosophies' behind the approaches. We focus here mainly on supervised learning and classification in particular.

13.2.1 Utility and Loss

To more fully specify a supervised problem we need to be clear what 'cost' is involved in making a correct or incorrect prediction. In a two class problem $\text{dom}(c) = \{1, 2\}$, we assume here that everything we know about the environment is contained in a model $p(x, c)$. Given a new input x^*, the optimal prediction also depends on how costly making an error is. This can be quantified using a loss function (or conversely a utility). In forming a *decision function* $c(x^*)$ that will produce a class label for the new input x^*, we don't know the true class, only our presumed distribution $p(c|x^*)$. The expected utility for the decision function is

$$U(c(x^*)) = \sum_{c^{true}} U(c^{true}, c(x^*))p(c^{true}|x^*) \tag{13.2.1}$$

and the optimal decision is that which maximises the expected utility.

Zero-one loss

A 'count the correct predictions' measure of prediction performance is based on the 'zero-one' utility (or conversely the *zero-one loss*):

$$U(c^{true}, c^*) = \begin{cases} 1 \text{ if } c^* = c^{true} \\ 0 \text{ if } c^* \neq c^{true} \end{cases} \tag{13.2.2}$$

For the two class case, we then have

$$U(c(x^*)) = \begin{cases} p(c^{true} = 1|x^*) \text{ for } c(x^*) = 1 \\ p(c^{true} = 2|x^*) \text{ for } c(x^*) = 2 \end{cases} \tag{13.2.3}$$

Hence, in order to have the highest expected utility, the decision function $c(x^*)$ should correspond to selecting the highest class probability $p(c|x^*)$:

$$c(x^*) = \begin{cases} 1 & \text{if} \quad p(c = 1|x^*) \geq 0.5 \\ 2 & \text{if} \quad p(c = 2|x^*) \geq 0.5 \end{cases} \tag{13.2.4}$$

In the case of a tie, either class is selected at random with equal probability.

General loss functions

In general, for a two-class problem, we have

$$U(c(x^*)) = \begin{cases} U(c^{true}=1, c^*=1)p(c^{true}=1|x^*) + U(c^{true}=2, c^*=1)p(c^{true}=2|x^*) & \text{for } c(x^*)=1 \\ U(c^{true}=1, c^*=2)p(c^{true}=1|x^*) + U(c^{true}=2, c^*=2)p(c^{true}=2|x^*) & \text{for } c(x^*)=2 \end{cases} \tag{13.2.5}$$

and the optimal decision function $c(x^*)$ chooses that class with highest expected utility.

One can readily generalise this to multiple-class situations using a *utility matrix* with elements

$$U_{i,j} = U(c^{true}=i, c^{pred}=j) \tag{13.2.6}$$

where the i, j element of the matrix contains the utility of predicting class j when the true class is i. Conversely one could think of a loss-matrix with entries $L_{ij} = -U_{ij}$. The expected loss with respect to $p(c|x)$ is then termed the *risk*.

In some applications the utility matrix is highly non-symmetric. Consider a medical scenario in which we are asked to predict whether or not the patient has cancer $\text{dom}(c) = \{\mathsf{cancer}, \mathsf{benign}\}$. If the true class is cancer yet we predict benign, this could have terrible consequences for the patient. On the other hand, if the class is benign yet we predict cancer, this may be less disastrous for the patient. Such asymmetric utilities can bias the predictions in favour of conservative decisions – in the cancer case, we would be more inclined to decide the sample is cancerous than benign, even if the predictive probability of the two classes is equal.

13.2.2 What's the catch?

In solving for the optimal decision function $c(x^*)$ above we are assuming that the model $p(c|x)$ is 'correct'. The catch is therefore that in practice :

- We typically *don't* know the correct model underlying the data – all we have is a dataset of examples $\mathcal{D} = \{(x^n, y^n), n = 1, \ldots, N\}$ and our domain knowledge.
- We want our method to perform well not just on a specifically chosen x^*, but any new input that could come along – that is we want it to *generalise* to novel inputs. This means we also need a model for $p(x)$ in order to measure what the expected performance of our decision function would be. Hence we require knowledge of the joint distribution $p(c, x) = p(c|x)p(x)$.

We therefore need to form a distribution $p(x, c|\mathcal{D})$ which should ideally be close to the true but unknown joint data distribution. Communities of researchers in machine learning form around different strategies to address the lack of knowledge about the true $p(c, x)$.

13.2.3 Using the empirical distribution

A direct approach to not knowing the correct model $p^{true}(c, x)$ is to replace it with the *empirical distribution*

$$p(x, c|\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N} \delta(x, x^n)\, \delta(c, c^n) \tag{13.2.7}$$

That is, we assume that the underlying distribution is approximated by placing equal mass on each of the points (x^n, c^n) in the dataset. Using this gives the empirical utility

$$\langle U(c, c(x)) \rangle_{p(c,x|\mathcal{D})} = \frac{1}{N}\sum_n U(c^n, c(x^n)) \tag{13.2.8}$$

or conversely the *empirical risk*

$$R = \frac{1}{N}\sum_n L(c^n, c(x^n)) \tag{13.2.9}$$