CHAPTER 13

Machine Learning Concepts

Machine learning is the body of research related to automated large-scale data analysis. Historically, the field was centred around biologically inspired models and the long term goals of much of the community are oriented to producing models and algorithms that can process information as well as biological systems. The field also encompasses many of the traditional areas of statistics with, however, a strong focus on mathematical models and also prediction. Machine learning is now central to many areas of interest in computer science and related large-scale information processing domains.

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 compact descriptions of the data. In both cases, one is interested in methods that generalise well to previously 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). We discuss first the basic characteristics of some learning frameworks before discussing supervised learning in more detail.

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 manner since formally specifying a rule that differentiates male from female faces is difficult. An alternative is to give examples faces and their gender labels and let a machine automatically 'learn' such a rule.

Definition 13.1 (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 novel input x^* the predicted output y^* is accurate. The pair (x^*, y^*) is not in $\mathcal{D}$ but assumed to be generated by the same unknown process that generated $\mathcal{D}$. 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

[1] For an $m \times n$ face image with elements F_{mn} we can form a vector by stacking the entries of the matrix.

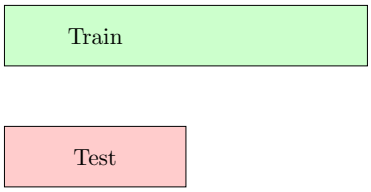


Figure 13.1: In training and evaluating a model, conceptually there are two sources of data. The parameters of the model are set on the basis of the train data only. If the test data is generated from the same underlying process that generated the train data, an unbiased estimate of the generalisation 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.

'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.

Example 13.1. 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 easy to formally specify 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.

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. 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 13.2 (Unsupervised learning)**.** Given a set of data $\mathcal{D} = \{x^n, n = 1, \ldots, N\}$ in unsupervised learning we aim to find 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 so that, from a probabilistic perspective, we are interested in modelling the distribution $p(x)$. The likelihood of the model to generate the data is a popular measure of the accuracy of the description.

Example 13.2. 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) 10000 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.

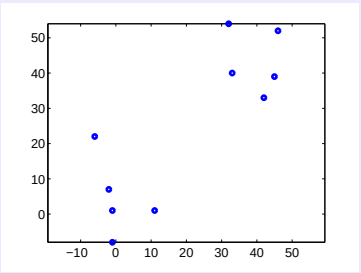
In the table each column represents the products bought by a customer (7 customer records with the first 6 of the 10,000 products are shown). A 1 indicates that the customer bought that item. We wish to find common patterns in the data, such as if someone buys coffee they are also likely to buy milk.

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 13.3 (Clustering)**.**

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

The table on the right represents a collection of unlabelled two-dimensional points. By simply eye-balling the data, we can see that there are two apparent clusters, one centred around (0,5) and the other around (35,45). A reasonable compact description of the data is that is has two clusters, one centred at (0,0) and one at (35,35), each with a standard deviation of 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 so that 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 environment and can distinguish the expected from the unexpected; this is an example of unsupervised learning. Detecting anomalous events in industrial processes (*plant monitoring*), engine monitoring and unexpected buying behaviour patterns in customers all fall under the area of unsupervised learning.

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 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 certain situations, an agent may be able to interact in some manner with its environment. This interaction can complicate but also enrich the potential for learning.

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 $p(x)$ is uninformative.

Reinforcement Learning In reinforcement learning an agent inhabits an environment in which it may take actions. 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 maximise the probability of obtaining a desired long term goal (long term survival, for example). 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(7.9.3), we will not discuss this topic further and refer the reader to specialised texts, for example [269].

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. This is a common issue in many examples since often gathering unlabelled data is cheap (taking photographs, for example). However, typically the labels are assigned by humans, which is expensive.

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 on supervised learning and classification in particular.

13.2.1 Utility and Loss

Given a new input x^*, the optimal prediction 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 then

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

and the optimal decision function $c(x^*)$ is that which maximises the expected utility,

$$c(x^*) = \underset{c(x^*)}{\text{argmax}}\, U(c(x^*)) \tag{13.2.2}$$

More generally, for a decision function $c(x|\theta)$ with parameters θ, the expected utility for decisions on a range of inputs described by the distribution $p(x^*)$ is

$$U(\theta) = \int \left\{ \sum_{c^{true}} U(c^{true}, c(x^*|\theta))p(c^{true}|x^*) \right\} p(x^*)dx^* = \langle U(c, c(x^*|\theta)) \rangle_{p(c,x^*)} \tag{13.2.3}$$

One may also consider equivalently a loss $L(c^{true}, c(x))$, for which the expected loss with respect to $p(c, x)$ is then termed the *risk*. The optimal decision function is then that which minimises the risk with respect to θ.

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*):

$$U(c^{true}, c^*) = \begin{cases} 1 \text{ if } c^* = c^{true} \\ 0 \text{ if } c^* \neq c^{true} \end{cases} \tag{13.2.4}$$

For the two class case, we then have

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

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

$$c(x^*) = \begin{cases} 1 & \text{if} \quad p(c = 1|x^*) > 0.5 \\ 2 & \text{if} \quad p(c = 2|x^*) > 0.5 \end{cases} \tag{13.2.6}$$

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

$$U(c(x^*)) = \begin{cases} U(c^{true} = 1, c^* = 1)p(c^{true} = 1|x^*) + U(c^{true} = 2, c^* = 1)p(c^{true} = 2|x^*) \text{ for } c(x^*) = 1 \\ U(c^{true} = 1, c^* = 2)p(c^{true} = 1|x^*) + U(c^{true} = 2, c^* = 2)p(c^{true} = 2|x^*) \text{ for } c(x^*) = 2 \end{cases} \tag{13.2.7}$$

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

$$U_{ij} = U(c^{true} = i, c^{pred} = j) \tag{13.2.8}$$

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 $L_{ij} = -U_{ij}$. 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 $\text{dom}(c) = \{\text{cancer}, \text{benign}\}$. 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 favour 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.

In solving for the optimal decision function $c(x|\theta)$ in equation (13.2.3) we are assuming that the model $p(c, x)$ is correct. However, in practice we typically don't know the correct model underlying the data – all we have is a dataset of examples $\mathcal{D} = \{(x^n, c^n), n = 1, \ldots, N\}$ and our domain knowledge. We therefore need to form a distribution $p(c, x|\mathcal{D})$ which should ideally be close to the true but unknown joint data distribution $p^{true}(c, x)$. Only then can our decisions be expected to generalise well to examples outside of the train data. Communities of researchers in machine learning form around different strategies to address the lack of knowledge about the true $p^{true}(c, x)$.

13.2.2 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*

$$p(c, x|\mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \delta(c, c^n) \delta(x, x^n) \tag{13.2.9}$$

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 expected utility

$$\langle U(c, c(x)) \rangle_{p(c,x|\mathcal{D})} = \frac{1}{N} \sum_{n} U(c^n, c(x^n)) \tag{13.2.10}$$

or conversely the *empirical risk*

$$R = \frac{1}{N} \sum_{n} L(c^n, c(x^n)) \tag{13.2.11}$$

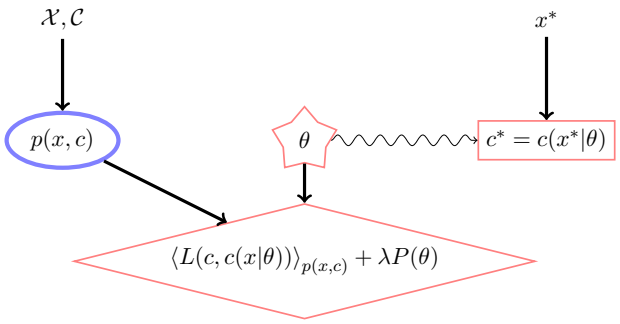


Figure 13.2: Empirical risk approach. Given the dataset $\mathcal{X}, \mathcal{C}$, a model of the data $p(x, c)$ is made, usually using the empirical distribution. For a classifier $c(x|\theta)$, the parameter θ is learned by minimising the penalised empirical risk with respect to θ. The penalty parameter λ is set by validation. A novel input x^* is then assigned to class $c(x^*|\theta)$, given this optimal θ.

Assuming the loss is minimal when the correct class is predicted, the optimal decision $c(x)$ for any input in the train set is given by $c(x^n) = c^n$. However, for any new x^* not contained in $\mathcal{D}$ then $c(x^*)$ is undefined. In order to define the class of a novel input, one may use a parametric function $c(x|\theta)$. For example for a two class problem $\text{dom}(c) = \{1, 2\}$, a linear decision function is given by

$$c(\mathbf{x}|\theta) = \begin{cases} 1 \text{ if } \boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_0 \geq 0 \\ 2 \text{ if } \boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_0 < 0 \end{cases} \tag{13.2.12}$$

If the vector input $\mathbf{x}$ is on the positive side of a hyperplane defined by the vector $\boldsymbol{\theta}$ and bias θ_0, we assign it to class 1, otherwise to class 2. (We return to the geometric interpretation of this in chapter(17)). The empirical risk then becomes a function of the parameters $\theta = \{\boldsymbol{\theta}, \theta_0\}$,

$$R(\theta|\mathcal{D}) = \frac{1}{N} \sum_n L(c^n, c(x^n|\theta)) \tag{13.2.13}$$

The optimal parameters θ are given by minimising the empirical risk with respect to θ,

$$\theta_{opt} = \underset{\theta}{\text{argmin}}\ R(\theta|\mathcal{D}) \tag{13.2.14}$$

The decision for a new datapoint x^* is then given by $c(x^*|\theta_{opt})$.

In this *empirical risk minimisation* approach, as we make the decision function $c(x|\theta)$ more flexible, the empirical risk goes down. However, if we make $c(x|\theta)$ too flexible we will have no confidence $c(x|\theta)$ will perform well on a novel input x^*. The reason for this is that a flexible decision function $c(x|\theta)$ is one for which, independent of x, the class label can change for only a small change in x. Such flexibility is great since it means that we will be able to find a parameter setting θ so that the train data is fitted well. However, since we are only constraining the decision function on the known training points, a flexible $c(x|\theta)$ may change rapidly as we move away from the train data, leading to poor generalisation. To constrain the complexity of $c(x|\theta)$ we may minimise the *penalised empirical risk*

$$R'(\theta|\mathcal{D}) = R(\theta|\mathcal{D}) + \lambda P(\theta) \tag{13.2.15}$$

where $P(\theta)$ is a function that penalises complex functions $c(x|\theta)$. The *regularisation constant*, λ, determines the strength of this penalty and is typically set by validation – see below. The empirical risk approach is summarised in fig(13.2).

For the linear decision function above, it is reasonable to penalise wildly changing classifications in the sense that if we change the input $\mathbf{x}$ by only a small amount we expect (on average) minimal change in the class label. The squared difference in $\boldsymbol{\theta}^\mathsf{T}\mathbf{x} + \theta_0$ for two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$ is $\left(\boldsymbol{\theta}^\mathsf{T}\Delta\mathbf{x}\right)^2$ where $\Delta\mathbf{x} \equiv \mathbf{x}_2 - \mathbf{x}_1$. By constraining the length of $\boldsymbol{\theta}$ to be small we limit the ability of the classifier to change class for only a small change in the input space. Assuming the distance between two datapoints is distributed according to an isotropic multivariate Gaussian with zero mean and covariance $\sigma^2\mathbf{I}$, the average squared change is $\left\langle \left(\boldsymbol{\theta}^\mathsf{T}\Delta\mathbf{x}\right)^2 \right\rangle = \sigma^2\boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$, motivating the choice of the Euclidean squared length of the parameter $\boldsymbol{\theta}$ as the penalty term, $P(\theta) = \boldsymbol{\theta}^\mathsf{T}\boldsymbol{\theta}$.

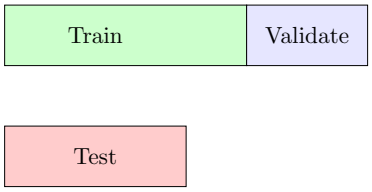


Figure 13.3: Models can be trained using the train data based on different regularisation parameters. The optimal regularisation parameter is determined by the empirical performance on the validation data. An independent measure of the generalisation performance is obtained by using a separate test set.

Algorithm 13.1 Setting regularisation parameters using cross-validation.

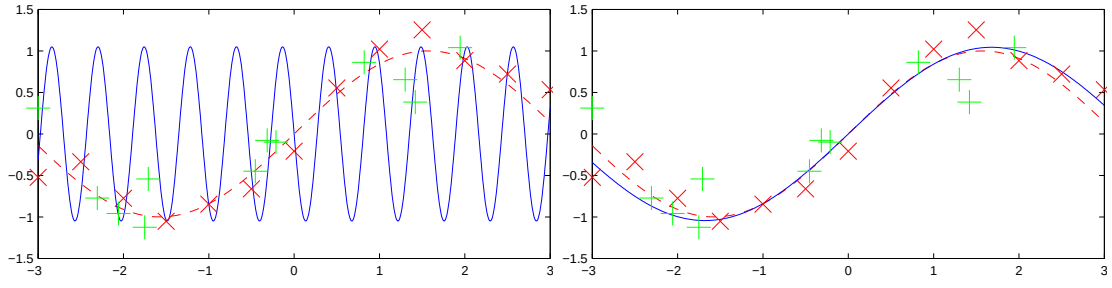
1: Choose a set of regularisation parameters $\lambda_1, \ldots, \lambda_A$
2: Choose a set of training and validation set splits $\left\{\mathcal{D}^i_{train}, \mathcal{D}^i_{validate}\right\}, i = 1, \ldots, I$
3: **for** $a = 1$ to A **do**
4: **for** $i = 1$ to I **do**
5: $\theta^i_a = \underset{\theta}{\text{argmin}} \left[R(\theta|\mathcal{D}^i_{train}) + \lambda_a P(\theta)\right]$
6: **end for**
7: $L(\lambda_a) = \frac{1}{I}\sum_{i=1}^{I} R(\theta^i_a|\mathcal{D}^i_{validate})$
8: **end for**
9: $\lambda_{opt} = \underset{\lambda_a}{\text{argmin}}\ L(\lambda_a)$

Validation

In penalised empirical risk minimisation we need to set the regularisation constant λ. This can be achieved by evaluating the performance of the learned classifier $c(x|\theta)$ on validation data $\mathcal{D}_{validate}$ for several different λ values, and choosing the λ which gave rise to the classifier with the best performance. It's important that the validation data is not the data on which the model was trained since we know that the optimal setting for λ in that case is zero, and again we will have no confidence in the generalisation ability.

Given a dataset $\mathcal{D}$ we split this into disjoint parts, $\mathcal{D}_{train}, \mathcal{D}_{validate}$, where the size of the validation set is usually chosen to be smaller than the train set. For each parameter λ_a one then finds the minimal empirical risk parameter θ_a. The optimal λ is chosen as that which gives rise to the model with the minimal validation risk, see fig(13.3). Using the optimal regularisation parameter λ, many practitioners retrain θ on the basis of the whole dataset $\mathcal{D}$.

In cross-validation the dataset is partitioned into training and validation sets multiple times with validation results obtained for each partition. Each partition produces a different training $\mathcal{D}^i_{train}$ and validation $\mathcal{D}^i_{validation}$ set, along with an optimal penalised empirical risk parameter θ^i_a and associated (unregularised) validation performance $R(\theta^i_a|\mathcal{D}^i_{validate})$. The performance of regularisation parameter λ_a is taken as the average of the validation performances over i. The best regularisation parameter is then given as that with the minimal average validation error, see algorithm(13.1). More specifically, in K-fold cross-validation the



(a) (b)

Figure 13.4: **(a)**: The unregularised fit ($\lambda = 0$) to training given by $\times$. Whilst the training data is well fitted, the error on the validation examples, denoted by $+$, is high. **(b)**: The regularised fit ($\lambda = 0.5$). Whilst the train error is high, the validation error is low. The true function which generated the noisy data is the dashed line; the function learned from the data is given by the solid line.

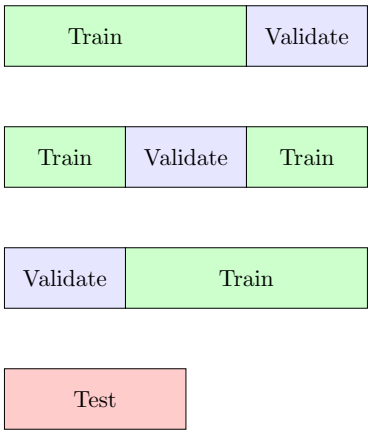


Figure 13.5: In cross-validation the dataset is split into several train-validation sets. Depicted is 3-fold cross-validation. For a range of regularisation parameters, the optimal regularisation parameter is found based on the empirical validation performance averaged across the different splits.

data $\mathcal{D}$ is split into K equal sized disjoint parts $\mathcal{D}_1, \ldots, \mathcal{D}_K$. Then $\mathcal{D}^i_{validate} = \mathcal{D}_i$ and $\mathcal{D}^i_{train} = \mathcal{D} \backslash \mathcal{D}^i_{validate}$. This gives a total of K different training-validation sets over which performance is averaged, see fig(13.5). In practice 10-fold cross-validation is popular, as is leave-one-out cross-validation in which the validation sets consist of only a single example.

Example 13.4 (Finding a good regularisation parameter)**.** In fig(13.4), we fit the function $a \sin(wx)$ to data, learning the parameters a and w. The unregularised solution fig(13.4a) badly overfits the data, and has a high validation error. To encourage a smoother solution, a regularisation term $E_{reg} = w^2$ is used. The validation error based on several different values of the regularisation parameter λ was computed, and $\lambda = 0.5$ gives the lowest validation error. The resulting fit to the data, fig(13.4b) is reasonable.

Benefits of the empirical risk approach

- In the limit of a large amount of training data the empirical distribution tends to the correct distribution.
- The discriminant function is chosen on the basis of minimal risk, which is the quantity we are ultimately interested in.
- The procedure is conceptually straightforward.

Drawbacks of the empirical risk approach

- It seems extreme to assume that the data follows the empirical distribution, particularly for small amounts of training data. More reasonable assumptions as to $p(x)$ would take into account likely x that could arise, not just those in the train data.
- If the loss function changes, the discriminant function needs to be retrained.
- Some problems require an estimate of the confidence of the prediction. Whilst there may be heuristic ways to evaluating confidence in the prediction, this is not inherent in the framework.
- When there are many penalty parameters, performing cross-validation in a discretised grid of the parameters becomes infeasible.
- During validation, many models are trained, and all but one subsequently discarded.

13.2.3 Bayesian decision approach

An alternative to using the empirical distribution is to first fit a model $p(c, x|\theta)$ to the train data $\mathcal{D}$. Given this model, the decision function $c(x)$ is automatically determined from the maximal expected utility (or minimal risk) with respect to this model, as in equation (13.2.7), in which the unknown $p(c^{true}|x)$ is replaced with $p(c|x, \theta)$.

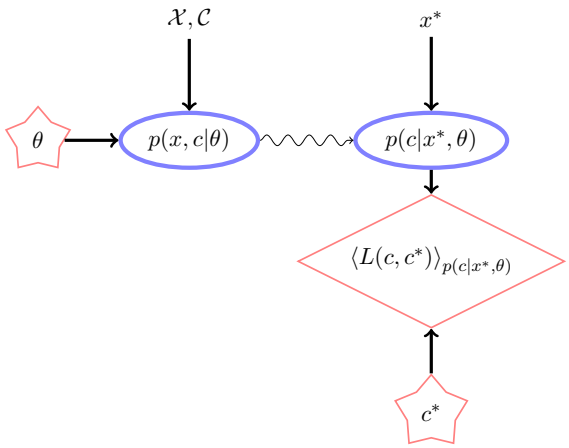


Figure 13.6: Bayesian decision approach. A model $p(x, c|\theta)$ is fitted to the data. After leaning the optimal model parameters θ, we compute $p(c|x, \theta)$. For a novel x^*, the distribution of the assumed 'truth', $p(c|x^*, \theta)$. The prediction (decision) is then given by that c^* which minimises the expected risk $\langle L(c, c^*)\rangle_{p(c|x^*,\theta)}$.

There are two main approaches to fitting $p(c, x|\theta)$ to data $\mathcal{D}$. We could parameterise the joint distribution using

$$p(c, x|\theta) = p(c|x, \theta_{c|x})p(x|\theta_x) \qquad \textit{discriminative approach} \tag{13.2.16}$$

or

$$p(c, x|\theta) = p(x|c, \theta_{x|c})p(c|\theta_c) \qquad \textit{generative approach} \tag{13.2.17}$$

We'll consider these two approaches below in the context of trying to make a system that can distinguish between a male and female face. The setup is that we have a database of face images in which each image is represented as a real-valued vector $\mathbf{x}^n, n = 1, \ldots, N)$, along with a label $c^n \in \{0, 1\}$ stating if the image is male or female.

Generative approach $p(\mathbf{x}, c|\theta) = p(\mathbf{x}|c, \theta_{x|c})p(c|\theta_c)$

For simplicity we use Maximum Likelihood training for the parameters θ. Assuming the data $\mathcal{D}$ is i.i.d., we have a log likelihood

$$\log p(\mathcal{D}|\theta) = \sum_n \log p(\mathbf{x}^n|c^n, \theta_{x|c}) + \sum_n \log p(c^n|\theta_c) \tag{13.2.18}$$

As we see the dependence on $\theta_{x|c}$ occurs only in the first term, and θ_c only occurs in the second. This means that learning the optimal parameters is equivalent to isolating the data for the male-class and fitting a model $p(\mathbf{x}|c = \mathsf{male}, \theta_{x|\mathsf{male}})$. We may similarly isolate the female data and fit a separate model $p(\mathbf{x}|c = \mathsf{female}, \theta_{x|\mathsf{female}})$. The class distribution $p(c|\theta_c)$ is set according to the ratio of males/females in the set of training data.

To make a classification of a new image $\mathbf{x}^*$ as either male or female, we use Bayes' rule:

$$p(c = \mathsf{male}|\mathbf{x}^*) = \frac{p(\mathbf{x}^*, c = \mathsf{male}|\theta_{x|\mathsf{male}})}{p(\mathbf{x}^*, c = \mathsf{male}|\theta_{x|\mathsf{male}}) + p(\mathbf{x}^*, c = \mathsf{female}|\theta_{x|\mathsf{female}})} \tag{13.2.19}$$

Based on zero-one loss, if this probability is greater than 0.5 we classify $\mathbf{x}^*$ as male, otherwise female. For a general loss function, we use this probability as part of a decision process, as in equation (13.2.7).

Advantages Prior information about the structure of the data is often most naturally specified through a generative model $p(x|c)$. For example, for male faces, we would expect to see heavier eyebrows, a squarer jaw, *etc.*

Disadvantages The generative approach does not directly target the classification model $p(c|x)$ since the goal of generative training is rather to model $p(x|c)$. If the data x is complex, finding a suitable generative data model $p(x|c)$ is a difficult task. Furthermore, since each generative model is separately trained for each class, there is no competition amongst the models to explain the data. On the other hand it might be that making a model of $p(c|x)$ is simpler, particularly if the decision boundary between the classes has a simple form, even if the data distribution of each class is complex, see fig(13.8).

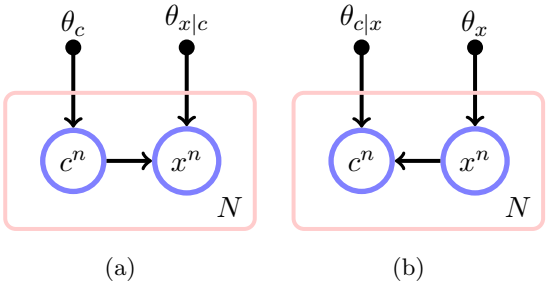


Figure 13.7: Two generic strategies for probabilistic classification. **(a)**: Class dependent generative model of x. After learning parameters, classification is obtained by making x evidential and inferring $p(c|x)$. **(b)**: A discriminative classification method $p(c|x)$.

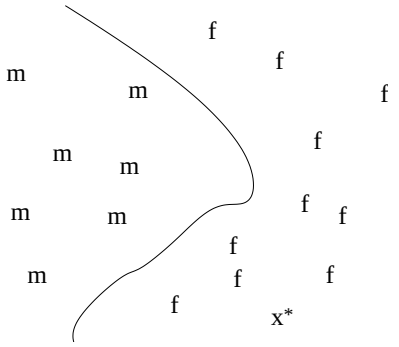


Figure 13.8: Each point represents a high dimensional vector with an associated class label, either male or female. The point $\mathbf{x}^*$ is a new point for which we would like to predict whether this should be male or female. In the generative approach, a male model $p(\mathbf{x}|\mathsf{male})$ generates data similar to the 'm' points. Similarly, the female model $p(\mathbf{x}|\mathsf{female})$ generates points that are similar to the 'f' points above. We then use Bayes' rule to calculate the probability $p(\mathsf{male}|\mathbf{x}^*)$ using the two fitted models, as given in the text. In the discriminative approach, we directly make a model of $p(\mathsf{male}|\mathbf{x}^*)$, which cares less about how the points 'm' or 'f' are distributed, but more about describing the boundary which can separate the two classes, as given by the line.

Discriminative approach $p(\mathbf{x}, c|\theta) = p(c|\mathbf{x}, \theta_{c|x})p(\mathbf{x}|\theta_x)$

Assuming i.i.d. data, the log likelihood is

$$\sum_n \log p(\mathcal{D}|\theta) = \sum_n \log p(c^n|\mathbf{x}^n, \theta_{c|x}) + \sum_n \log p(\mathbf{x}^n|\theta_x) \tag{13.2.20}$$

The parameters are isolated in the two terms so that Maximum Likelihood training is equivalent to finding the parameters of $\theta_{c|x}$ that will best predict the class c for a given training input x. The parameters θ_x for modelling the data occur only in the second term above, and setting them can therefore be treated as a separate unsupervised learning problem. This approach consequently isolates modelling the *decision boundary* from modelling the input distribution, see fig(13.8).

Classification of a new point $\mathbf{x}^*$ is based on

$$p(c|\mathbf{x}, \theta^{opt}_{c|x}) \tag{13.2.21}$$

As for the generative case, this approach still learns a joint distribution $p(c, x) = p(c|x)p(x)$ which can be used as part of a decision process if required, equation (13.2.7).

Advantages The discriminative approach directly addresses finding an accurate classifier $p(c|x)$ based on modelling the decision boundary, as opposed to the class conditional data distribution in the generative approach. Whilst the data from each class may be distributed in a complex way, it could be that the decision boundary between them is relatively easy to model.

Disadvantages Discriminative approaches are usually trained as 'black-box' classifiers, with little prior knowledge built in as to how data for a given class might look. Domain knowledge is often more easily expressed using the generative framework.

Features and preprocessing

It is often the case that in discriminative training, transforming the raw input x into a form that more directly captures the relevant label information can greatly improve performance. For example, in the male-female classification case, it might be that building a classifier directly in terms of the elements of the face vector $\mathbf{x}$ is difficult. However, using 'features' which contain geometric information such as the distance between eyes, width of mouth, *etc.* may make finding a classifier easier. In practice data is also often preprocessed to remove noise, centre an image *etc.*

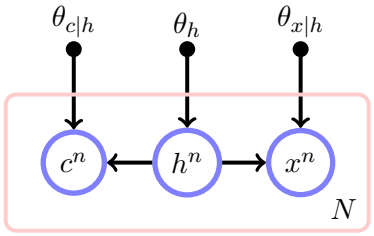


Figure 13.9: A strategy for semi-supervised learning. When c^n is missing, the term $p(c^n|h^n)$ is absent. The large amount of training data helps the model learn a good lower dimension/compressed representation h of the data x. Fitting then a classification model $p(c|h)$ using this lower dimensional representation may be much easier than fitting a model directly from the complex data to the class, $p(c|x)$.

Benefits of the Bayesian decision approach

- This is a conceptually 'clean' approach, in which one tries ones best to model the environment, independent of the subsequent decision process. In this case learning the environment is separated from the effect this will have on the expected utility.
- The decision c^* for a novel input x^* can be a highly complex function of x^* due to the maximisation operation.
- If $p(x, c|\theta)$ is the 'true' model of the data, this approach is optimal.

Drawbacks of the Bayesian decision approach

- If the environment model $p(c, x|\theta)$ is poor, the prediction c^* could be highly inaccurate since modelling the environment is divorced from prediction.
- To avoid fully divorcing the learning of the model $p(c, x|\theta)$ from its effect on decisions, in practice one often includes regularisation terms in the environment model $p(c, x|\theta)$ which are set by validation based on an empirical loss.

Hybrid generative-discriminative approaches

One could use a generative description, $p(x|c)$, building in prior information, and use this to form a joint distribution $p(x, c)$, from which a discriminative model $p(c|x)$ may be formed, using Bayes' rule. Specifically, we can use

$$p(c|x, \theta) = \frac{p(x|c, \theta_{x|c})p(c|\theta_c)}{\sum_c p(x|c, \theta_{x|c})p(c|\theta_c)} \tag{13.2.22}$$

Subsequently the parameters $\theta = (\theta_{x|c}, \theta_c)$, for this hybrid model can be found by maximising the probability of being in the correct class. A separate model is learned for $p(x|\theta_x)$. This approach would appear to leverage the advantages of both the discriminative and generative frameworks since we can more readily incorporate domain knowledge in the generative model $p(x|c, \theta_{x|c})$ yet train this in a discriminative way. This approach is rarely taken in practice since the resulting functional form of the likelihood depends in a complex manner on the parameters. In this case no parameter separation between θ_c and $\theta_{x|c}$ occurs (as was previously the case for the generative and discriminative approaches).

13.2.4 Learning lower-dimensional representations in semi-supervised learning

One way to exploit a large amount of unlabelled training data to improve classification is to first find a lower dimensional representation h of the data x. Based on this, the mapping from h to c may be rather simpler to learn than a mapping from x to c directly. To do so we can form the likelihood using, see fig(13.9),

$$p(\mathcal{C}, \mathcal{X}, \mathcal{H}|\theta) = \prod_n \left\{p(c^n|h^n, \theta_{c|h})\right\}^{\mathbb{I}[c^n \neq \emptyset]} p(x^n|h^n, \theta_{x|h})p(h|\theta_h) \tag{13.2.23}$$

and then set any parameters for example by using Maximum Likelihood

$$\theta^{opt} = \underset{\theta}{\text{argmax}} \sum_{\mathcal{H}} p(\mathcal{C}, \mathcal{X}, \mathcal{H}|\theta) \tag{13.2.24}$$