APTS: Statistical Machine Learning Preliminary Material

Louis J.M. Aslett
Department of Mathematical Sciences
Durham University

Contents

Welcome to the Statistical Machine Learning module on the Academy for PhD Training in Statistics.

The 2023–24 module is scheduled to run from 1st–5th September 2025 at the University of Southampton. Any administrative queries regarding the week should be directed to the APTS organisers.

This preliminary material is provided to offer a little background/motivation for the course itself and to introduce you to thinking about the core machine learning problem via a simple toy example.

Section 1 is selectively reproduced from the notes for the main course and provides the background/motivation. Section 2 and 3 explain what topics you should be familiar with.

Section 4 is a very gentle technical excursion into predictive accuracy, using a toy example. Note that it is probable you are already familiar with all the topics discussed, especially if you have attended some of the earlier APTS weeks: the intention is to provide a very simple example that you can understand deeply and use as a mental picture when studying the course itself, since from the outset we will embark on a more abstract and general learning theoretic development.

However, naturally people embarking on a PhD in Statistics often come from multifarious backgrounds, so it is possible this material also highlights a gap you may wish to fill to ensure you get the most from the course.

You can download these notes in PDF, but note that you lose access to the live code features of this web version which are critical for this preliminary material.

1 The module

Aims: This module introduces students to modern supervised machine learning methodology and practice, with an emphasis on statistical and probabilistic approaches in the field. The course seeks to balance theory, methods and application, providing an introduction with firm foundations that is accessible to those working on applications and seeking to employ best practice. There will be exploration of some key software tools which have facilitated the growth in the use of these methods across a broad spectrum of applications and an emphasis on how to carefully assess machine learning models.

Learning Outcomes: Students following this module will gain a broad view of the supervised statistical machine learning landscape, including some of the main theoretical and methodological foundations. They will be able to appraise different machine learning methods and reason about their use. In particular, students completing the course will gain an understanding of how to practically apply these techniques, with an ability to critically evaluate the performance of their models. Students will also have an insight into the extensive software libraries available today and their use to construct a full machine learning pipeline.

Prerequisites

To undertake this module, students should have:

- at least one undergraduate level course in probability and in statistics;
- standard undergraduate level knowledge of linear algebra and calculus;
- solid grasp of statistical computing in R;
- knowledge of statistical modelling, including regression modelling (eg. APTS Statistical Modelling course);
- some basic understanding of optimisation methods beneficial, but not essential.

As preparatory reading, the enthusiastic student may choose to browse An Introduction to Statistical Learning (James et al., 2013) (freely and legally available online), which covers some of the topics of the course at a more elementary and descriptive level.

Textbooks at roughly the level of the course include:

- The Elements of Statistical Learning (Hastie et al., 2009)

- Pattern Recognition and Machine Learning (Bishop, 2006)
- Machine Learning: A Probabilistic Perspective (Murphy, 2012)

1.1 Live code

In this course, we consider code to be a first class citizen in your learning: to be able to apply or understand the content it is often helpful to be able to see and play with illustrative code. We will be using the statistician's favourite tool, R (R Core Team, 2021)! Some code, especially if it is long-running, will simply be displayed inline like the following:

```
x <- rnorm(10)
mean(x)
```

However, where a fast running toy example is possible, this will be used and provided in "live code" blocks, where you can make changes and re-run to observe the effect. These appear like this:

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

You can run the code inline by clicking the *"Run code!"* button without leaving these notes and losing your flow. Also, since the code is remotely executed this will work even on mobile devices like an iPad or iPhone where R is not supported.

Where there is more than one live block on the page, you can link them together into a persistent session, akin to an RStudio Notebook or Jupyter Notebook, by enabling the "Persistent" toggle. Note that like those other notebook environments, the order in which you run blocks matters, so that if you run a second block trying to access a variable `x` before the first block assigns to it you will encounter an error. Therefore, when needed you can evaluate all preceding blocks with the *"Run previous"* button which appears when you enable a persistent session. To see this, click the "Persistent" toggle on and try running the next block: note this results in an error, because the first code block has not been run with persistence enabled. Therefore, run the first block before trying this one again, which should result in the value of `x` printing correctly:

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

It is possible to include plots and other more complex output too. To see this, modify the code above to add `hist(x)` on line 2, click the *"Run code!"* button again and either *"Zoom"* or scroll to see the histogram (note, you don't need to rerun the first block this time, because your session now has `x` defined).

These preliminary notes do not require a persistent session, so turn that toggle off again before continuing.

The server running these code chunks is a shared and public resource, so please be considerate. You should also assume that anything you run may be publicly visible, so *do not run anything confidential*!

1.2 What *is* "Statistical Machine Learning"?

As Larry Wasserman (2014) puts it, "Statistics is the science of learning from data. Machine Learning (ML) is the science of learning from data. These fields are identical in intent

although they differ in their history, conventions, emphasis and culture."

This course is primarily concerned with supervised machine learning which, despite the modern buzz, has been around for a considerable time in other guises. An early related moniker was *"pattern recognition"* and indeed, even as early as 1968 one can find review papers packed with methods that are familiar to the modern practitioner of machine learning (Nagy, 1968). It was recognised even then that this was something of a distinct field that draws on many others, including especially statistics.

Around this time there was also cogitation within the statistics community, with some encouraging a broadening of scope from the focused mathematical ideals of the day: in his treatise "The Future of Data Analysis", John Tukey (1962) advocated passionately for statistics to shift focus. Jerome Friedman (1998) revived the debate about whether statistics should embrace the other blossoming disciplines seeking to learn from data, with pattern recognition being joined by "data mining", "machine learning" and "artificial intelligence" in a list of related fields sharing similar goals. Contemporaneously, Leo Breiman (2001) was more forthright, arguing clearly in favour of statistics embracing algorithmic methods and a focus on predictive accuracy which is common in machine learning. Most recently, Brad Efron (2020) wrote a compelling account relating modern machine learning methods to traditional statistical approaches.

However, there are certainly critics:

> "Machine learning is statistics minus checking of models and assumptions."
>
> *Brian Ripley, useR! 2004, Vienna*

Although this epigrammatic characterisation of machine learning is oft humorously quoted and perhaps a little unfair, it does reflect the fact that a machine learning analysis can exhibit a markedly different feel to other forms of statistical modelling. A machine learner may (perhaps unwisely) retort that many black box algorithmic methods do not have a specific model and make minimal assumptions.

It is interesting to read the above references for some historical context, but it should be uncontroversial to say that today there are sufficiently many statisticians working on machine learning theory, methods, and applications, that statistics certainly has a voice in the machine learning community. Indeed, it has "had a stimulating effect on traditional theory and practice" (Efron, 2020).

Suffice to say, we will therefore avoid attempting to dichotomise statistics and machine learning, instead attempting only a loose characterisation of what we mean by supervised machine learning for the purposes of this course: a collection of modelling approaches predominantly concerned with accurate predictive modelling, frequently employing algorithmic or black-box methodology which does not necessarily have interpretable parametrisations, and where parameter uncertainty is often of secondary concern or neglected. It is a subtle distinction, but it does matter that supervised machine learning is concerned primarily with prediction (whilst statistics is broader and encompasses explanatory modelling, estimation and attribution too (Efron, 2020; Shmueli, 2010))

What therefore of the "Statistical" prefix in the course title? From the perspective of a statistician, an unfortunate feature of *some* (not all) corners of machine learning practice is a failure to fully evaluate the uncertainty of predictions produced by these methods, a

disregard for proper scoring rules, or a contentment to set aside a mathematical treatment of the underpinning theory and methods. As such, the prefix "Statistical" is to highlight the emphasis we will place upon uncertainty quantification in predictive modelling and a desire to develop some mathematical understanding of foundations where it helps.

Work attempting to trace some of the the historical development and coexistence of machine learning and statistics are in Malik (2020) and Jones (2018). The Introduction in Vapnik (1998) argues the historical branching of machine learning theory from its statistics origins happened through four key discoveries in the 1960s-80s.

1.3 Notation

As far as possible, the whole course will use a consistent notation, with any deviation from it explained at the time.

- Scalars: $x, x_i, x_{ij}, y_i, \beta_i, \ldots$
- Vectors: $\mathrm{x}, \mathrm{y}, \boldsymbol{\beta}, \ldots$
  - follow the standard convention that all are column vectors
  - transpose x^T is row vector
  - x_i indicates a vector, the elements of which are x_{ij}
- Matrices: $\mathrm{X}, \mathrm{Y}, \ldots$
  - matrix transpose: X^T
  - i-th row entries: x_i (as column vector)
  - j-th column entries: $\mathrm{x}_{\cdot j}$
  - (i, j)-th element: x_{ij}
- Random variables: $X, Y, \varepsilon, \ldots$
  - clear from context whether a random scalar/vector/matrix/...
  - clear from context whether Greek letters are random variables
  - the probability measure associated with a random variable X is π_X.
- Spaces: $\mathcal{X}, \mathcal{Y}, \mathbb{R}, \mathbb{Z}, \mathbb{R}^d = \underbrace{\mathbb{R} \times \cdots \times \mathbb{R}}_{d \text{ times}}, \ldots$
- Estimator: denoted by a hat, $\hat{f}(\cdot), \hat{\boldsymbol{\beta}}, \ldots$
- Functions:
  - $\mathbb{1}\{A\}$ is the indicator function for A being true.
  - if $f(x)$ has vector valued output, then $f_j(x)$ denotes the j-th component of the output.
  - where necessary an arbitrary function, $f(x)$, will be distinguished from a probability density function (pdf), $f_X(x)$, by the presence or absence of a random variable subscript.
  - the cumulative distribution function (cdf) is denoted $F_X(x)$.
- Other:
  - $A \coloneqq B$ reads "A is defined to be B".

- For a finite set $\mathcal{I}$, $|\mathcal{I}|$ denotes the cardinality of the set.

1.4 Problem setting

We first set out the general problem statement and the usual approach to modelling adopted in a supervised machine learning setting.

The most common supervised machine learning problems fall broadly under three types (Vapnik, 1998):

- **Regression** models a *quantitative* outcome.
  - What value is a house based on geographic/house information;
  - How long until a patient is discharged from hospital?
- **Classification** models a *qualitative* outcome.
  - Medics predicting a disease from test results;
  - Is the email just sent to my address spam?
  - Bank predicting if borrower will default;
  - Identifying a number from image of handwritten value.
- **Density estimation** models a full *probability distribution*.

There are of course specialisations within these relating to the nature of the data, so the first regression example is a spatial regression problem, whilst the second is a survival regression problem, etc.

Supervised learning starts from the premise that we have access to a set of n observations of:

- *features* / *predictors* from some space $\mathcal{X}$, eg:
  - $\mathcal{X} \subset \mathbb{R}^d$
  - or, $\mathcal{X}$ can be a tensor in deep learning
- and corresponding *outcomes* / *responses* / *targets* from some space $\mathcal{Y}$, eg:
  - $\mathcal{Y} \subset \mathbb{R} \implies$ regression;
  - in particular, $\mathcal{Y} = \{1, \ldots, g\}$ where $g \geq 2 \implies$ classification, with each qualitative outcome assigned a numeric label;
  - or, for $g = 2$ we often[1] take $\mathcal{Y} = \{0, 1\}$
  - or, $\mathcal{Y}$ can be a tensor in deep learning

The full dataset we use to 'learn' is thus $\mathcal{D} = \big((\mathrm{x}_1, y_1), \ldots, (\mathrm{x}_n, y_n)\big) \subset (\mathcal{X} \times \mathcal{Y})^n$, where x_i is a vector of length d,

$$\mathrm{x}_i = (x_{i1}, \ldots, x_{id})^T \in \mathcal{X}$$

We denote all observations of a single feature as $\mathrm{x}_{\cdot j}$,

[1] You will encounter some texts which assign ± 1 rather than $\{0, 1\}$ as the response in binary classification problems.

$$\mathrm{x}_{\cdot j} = (x_{1j}, \ldots, x_{nj})^T$$

With this setup, we proceed to specify a model by assuming there is a functional relationship connecting features and response, together with an error/uncertainty process. Our objective is to learn this 'true' but unknown function. An example setting falling under each type above is:

- **Regression:** eg we take $f : \mathcal{X} \to \mathcal{Y}$ directly, often with an additive noise assumption so that
$$Y = f(\mathrm{x}) + \varepsilon$$
where ε is a random error term. If the error is assumed to have zero mean, the function f is the conditional expectation,
$$\mathbb{E}[Y \mid X = \mathrm{x}] = f(\mathrm{x})$$

- **Classification:**
  - *probabilistic classifier*, eg: we take $f : \mathcal{X} \to [0,1]^g$ as connecting features to the probability of each response in a categorical distribution,
  $$Y \mid X = \mathrm{x} \sim \text{Categorical}\big((p_1, \ldots, p_g) = f(\mathrm{x})\big)$$
  where $\sum_i p_i = 1$, with $p_i = \mathbb{P}(Y = i \mid X = \mathrm{x})$.

  Note in the binary case with $\mathcal{Y} = \{0, 1\}$,
  $$Y \mid X = \mathrm{x} \sim \text{Bernoulli}\big(p = f(\mathrm{x})\big)$$
  so that
  $$p = \mathbb{P}(Y = 1 \mid X = \mathrm{x}) = \mathbb{E}[Y \mid X = \mathrm{x}] = f(\mathrm{x})$$
  giving a natural correspondence with the regression setting.
  - *scoring classifier*, eg: sometimes there is no attempt to directly model the probability above. Instead, a real-valued score with no formal interpretation is assigned to each response label, the intention being that the highest scoring is (in some loosely defined sense) 'most likely'. In other words in the binary case, $f : \mathcal{X} \to \mathbb{R}$ such that
  $$y = \begin{cases} 0 & \text{if } f(\mathrm{x}) \leq 0 \\ 1 & \text{if } f(\mathrm{x}) > 0 \end{cases}$$

- **Density estimation:** eg conditional modelling of
$$Y \mid X = x \sim \mathbb{P}(Y \mid X = x)$$
possibly by jointly modelling $\mathbb{P}(X, Y)$ with some probability density function $f_{XY}(\mathrm{x}, y)$ (note a density, here, and function, above, are distinguished by the presence or absence of the random variable subscript).

In some sense the last of these subsumes the former two (and more) and is the most general setting: indeed, in all cases we take there to be some true but unknown probability measure, π_{XY}, which generates the data and usually assume[2] that each observation (x_i, y_i) is an independent and identically distributed (iid) realisation from this measure.

Framing the problem as that of density estimation via a full generative probability model may be most familiar and comfortable for a statistician. Indeed, thinking of the classification problem under this formulation leads to a collection of methods termed *generative approaches*, whereby a model is constructed for $X \mid Y = i$ (or the joint) and prediction made by a simple application of Bayes' Theorem. This is in contrast to direct modelling of $Y \mid X = \mathrm{x}$ which is often advocated and these are termed *discriminative approaches*. However, in reality neither approach universally dominates (Ng and Jordan, 2001).

As alluded to above, in a machine learning setting nearly all the focus of this modelling exercise is for prediction of future observations. In these future settings, it is assumed we will have access to the predictors, x, but not the response, y, and we seek a model which can generalise well to unseen data.

We will typically approach these problems by directly estimating $\hat{f}(\cdot) \approx f(\cdot)$, though in the classification case some methods instead only attempt to directly model the response and not the probabilistic estimate (these are somewhat antithetical to our statistical predilection). We will need to select some class of models to which we believe the true function belongs, $f \in \mathcal{F}$, which may be parametrised by some parameter β, $f(\cdot \mid \beta)$. You will see some machine learning literature refer to this space of models as the *hypothesis space*.

2 Probability essentials

You should ensure that you are comfortable with all the elementary results in probability theory, including notions of independence, mutual exclusivity, conditional probability; together with the standard methods of manipulating probability statements, including decomposing unions and intersections of events, event complements, the law of total probability and of course Bayes' Theorem.

You should also be comfortable with notions of event probabilities given both discrete and continuous probability distributions. If any of this background is in doubt, referencing Chapters 1–4 of DeGroot and Schervish (2012) would provide sufficient coverage.

In a slight abuse of terminology we will interchangeably refer to π_X as the measure or distribution of the random variables X, and denote the standard probability density function as $f_X(x)$, the subscript distinguishing this density from a standard function.

It will be of particular importance to be comfortable with interpreting and manipulating expectations of random variables. The vast majority of both theoretical and practical quantities of interest in statistical machine learning are expressed as expectations in one form or another. Given its centrality to learning theory, in this course we will be particularly careful to be explicit about what distribution every expectation is with respect to (especially as I have observed this being a source of confusion among students). The standard

[2] Time series data being a notable exception, among others.

definition[3] of expectation will therefore be written as:

$$\begin{aligned}\mathbb{E}_X\left[g(X)\right] &\coloneqq \int_{\mathcal{X}} g(x)\, d\pi_X \\ &= \int_{\mathcal{X}} g(x) f_X(x)\, dx\end{aligned}$$

with the subscript on $\mathbb{E}$ making clear that here the expectation is with respect to the random variable X, whose measure is π_X and with density $f_X(x)$. The first line is always true, the second line naturally only applying if the corresponding density function exists in a continuous setting.

3 Regression

In these preliminary notes, we will explore informally some of the concepts you will tackle rigorously in the course. We'll do that exploration using standard linear regression, since you should be familiar with this already, and we will introduce further specialised machine learning models during the course.

You may have encountered linear regression in detail during earlier APTS courses (especially Statistical Modelling, Ogden et al., 2021) or your previous studies. Standard linear regression can of course address the general machine learning regression problem set out above, with $\mathcal{X} \subseteq \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ and with the assumption that additive noise is normally distributed,

$$Y = f(\mathrm{x}) + \varepsilon, \qquad \varepsilon \sim \mathrm{N}(\mu = 0, \sigma^2)$$

Data is usually arranged in a design matrix $\mathrm{X} \in \mathbb{R}^{n \times d}$ for the features, with observations in rows and features in columns, and as a vector of responses $\mathrm{y} \in \mathbb{R}^n$. Then, we write $f(\mathrm{x}) = \mathrm{X}\boldsymbol{\beta}$ with the regression coefficients $\boldsymbol{\beta} \in \mathbb{R}^d$ parameterising the model. The per-observation residuals are simply $\varepsilon_i = y_i - \mathrm{x}_i^T\boldsymbol{\beta}$.

Given all the above, there is a closed form solution to the maximum likelihood estimation of the coefficients:

$$\hat{\boldsymbol{\beta}} = \left(\mathrm{X}^T\mathrm{X}\right)^{-1}\mathrm{X}^T\mathrm{y}$$

You should have encountered the duality whereby this is also the solution to the "least squares" problem: that is, where we ignore the probabilistic model on ε and instead simply seek to minimise $\sum_{i=1}^n \varepsilon_i^2$. We will see during the course that many models in machine learning which lack a formal probabilistic structure essentially rely on something akin to this second approach to the problem, an idea called *empirical risk minimisation*, which we will cover.

[3] Strictly speaking this is not the definition of expectation, but rather is the so-called *law of the unconscious statistician* which can be proved from the rigorous definition of expectation.

Armed with a fitted model, a statistician's first reflex may be to check the model assumptions (residual Normality and homoskedasticity, linearity, etc) and then to examine the significance of coefficients (or their posteriors in a Bayesian analysis). It *is* of course still important to do these things if you are fitting a probabilistic machine learning model, but in reality the ultimate litmus test for evaluating the model in a machine learning setting will typically be assessments of its predictive accuracy. Using the average of the squared residuals may be a simplest way to do so in this example.

4 Toy predictive regression

Note that "toy" is not meant disparagingly here: it is simply an indication that we want to use a sufficiently simple example that nothing is obfuscated and the core message can be understood easily. The core message we aim to unpack here is the subtle variations in the definition of prediction errors that we will study in some depth during the course, since these are all closely related and often treated a little loosely at undergraduate level.

4.1 The model

Therefore, let us consider the *very* simple setting of univariate regression modelling. Imagine that there is a true data generating process as follows:

$$\begin{aligned} X &\sim \text{Unif}([-0.5, 1]) \\ f(x) &= \frac{1}{2}x^2 - x + 1 \\ (Y \mid X = x) &\sim \text{N}(\mu = f(x), \sigma = 0.5) \end{aligned}$$

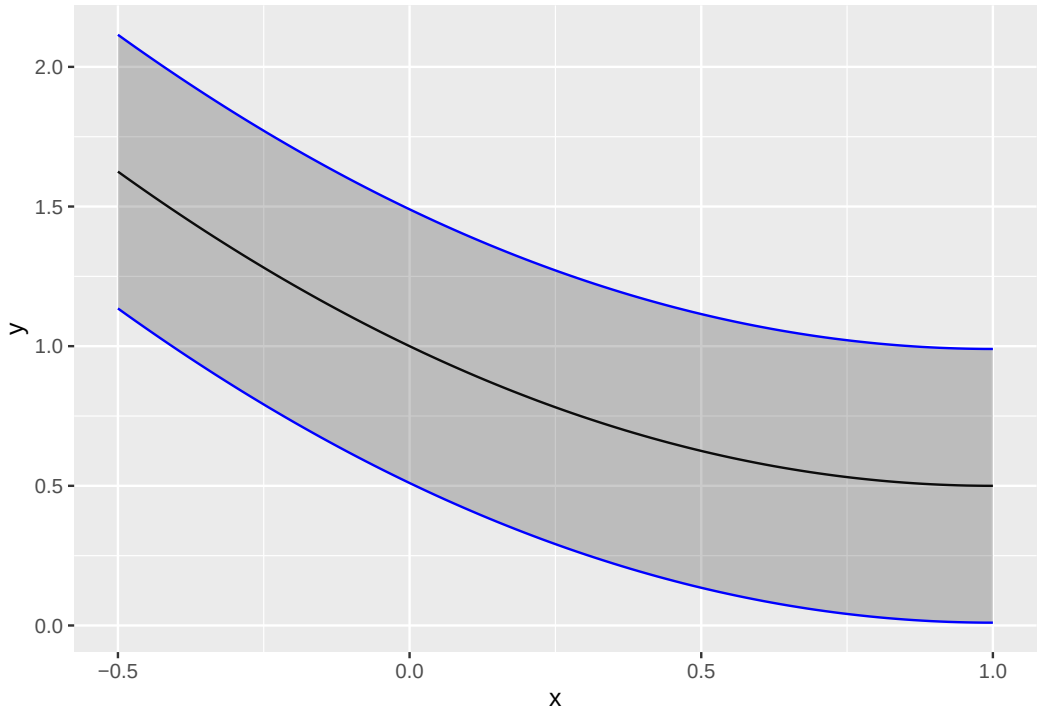
which we use to simulate a dataset of size n, $\mathcal{D}_n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, by iid sampling.

Recall that this *does* fall within the linear modelling framework, since the above model is linear in the parameters. Put simply, if we define the variable $z \coloneqq x^2$ then $\frac{1}{2}x^2 - x + 1 = \frac{1}{2}z - x + 1$ which is a linear model in the variables x, z. In practice this is achieved by creating a design matrix:

$$\mathrm{X} \coloneqq \begin{pmatrix} 1 & x_1 & x_1^2 \\ & \vdots & \\ 1 & x_n & x_n^2 \end{pmatrix}$$

Of course, if we fail to include the square transformation then a standard linear model using only x_i directly would be misspecified for this setting. In principle we can include higher powers and other transformations too in the same fashion, by adding additional columns to the design matrix.

Under the above model, 95% of observations would be expected to fall within the shaded grey region on the following plot, with the black line showing the conditional expectation of Y given a particular value x.



To highlight a core concern in machine learning, we will attempt to fit two models to a sample of simulated data:

$$\hat{f}^{(1)}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$
$$\hat{f}^{(2)}(x) = \hat{\beta}_0 + \sum_{j=1}^{5} \hat{\beta}_j x^j$$

4.2 Apparent error

Try running the following code and examine the output. It will simulate $n = 30$ observations from this model and fit $\hat{f}^{(1)}(x)$ (ie the misspecified linear model) as well as $\hat{f}^{(2)}(x)$ (ie the overparameterised linear model). It will then compute the mean square error in each case.

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

Imagine that we actually knew the true model exactly. Due to the Normally distributed error process we would never be able to make exact predictions, but what is the lowest mean square error we could hope to achieve? You may have seen before that predicting the mean value (ie the fitted line value) is the best option when minimising mean square error (MSE), in which case we would have:

$$\mathbb{E}_{Y \mid X=x}\left[\left(Y-\hat{f}(x)\right)^2\right]=\int_{\mathbb{R}}\left(Y-\hat{f}(x)\right)^2 \, d\pi_{Y \mid X=x}$$

where $\pi_{Y \mid X=x}$ is simply Normal with mean $f(x)$. In other words, the above integral is just the central second moment of the Normal distribution, in this case $\sigma^2 = 0.25^2 = 0.0625$.

With the random seed used in the live code above, you should have found the MSE of model 2 is 0.0478. Was it simply a fluke that we appear to have a better MSE under model 2 than under the true model? Remove the `set.seed()` line above and rerun a few times to see what MSE you achieve: you should find that most (though not all) of the time it is below 0.0625, so apparently better than knowing the truth.

Of course, this is clearly not the case, because we actually computed what is referred to as the *training* or *apparent* error: that is, the error made by the model when predicting the same data that was used to fit it. As we will see in the course (and you may have seen before), the apparent error is a *biased* estimate of the true error you would expect to see when predicting future responses y, given only x.

4.3 Generalisation error

4.3.1 Fixed design

Since this is a simulation experiment, we can actually explore what happens when we randomise the responses again, according to the model.

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

The above shows that the error estimate in both cases was optimistic, particularly so for $\hat{f}^{(2)}(x)$, so that both appear to have roughly similar performance now. However, what is this error quantity? This is what we call the estimated *fixed design* generalisation MSE. In other words, we are assessing the model's ability to predict new responses y at the same values of x that were used to fit the model. This is a quantity that is very frequently studied in statistical modelling because it is more tractable to compute (the marginal distribution π_X not needing to be accounted for).

4.3.2 Random design

However, if we are building a machine learning model because we are primarily focused on prediction, it is not unreasonable to actually be interested in how the model generalises to new feature values, not just new responses at the same design points. What we want is the *random design* generalisation MSE:

$$\mathbb{E}_{XY}\left[\left(Y-\hat{f}(X)\right)^2\right]$$

Note that here $\hat{f}$ is the fixed fitted model (either model 1 or model 2) and the randomness is entirely in the values being predicted under the fixed model.

Since we are running a toy simulation example, we can again estimate this quantity, now re-simulating both X and Y (this amounts to a Monte Carlo estimate of the integral in the expectation).

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

Interestingly, this time the models behave differently: $\hat{f}^{(1)}(x)$ seems to generalise outside the training observation feature values better than $\hat{f}^{(2)}(x)$, whose MSE has got worse still.

4.4 Expected error

Of course, crucially this may be 'luck' for this particular training data, $\mathcal{D}_{30}$ (which has been held fixed across the different live code boxes by the fixed seed). When working on an applied problem, of course in some sense the only performance which matters is the performance of the model fitted to the data we have in hand. However, when we are developing new methodology or are interested in the behaviour of a model in general, we may be more interested in how it performs across many different sets of training data from the same problem. That is, we want to really take a further expectation with respect to the training data:

$$\mathbb{E}_{D_n}\left[\mathbb{E}_{XY}\left[\left(Y - \hat{f}(X)\right)^2\right]\right]$$

where $D_n \coloneqq ((X_1, Y_1), \ldots, (X_n, Y_n))$ is a random variable denoting a sample of n training observations, whose joint distribution is $\pi_{XY}^n \coloneqq \underbrace{\pi_{XY} \times \cdots \times \pi_{XY}}_{n \text{ times}}$, noting that $\hat{f}$ depends on the realisation of D_n, that is $\mathcal{D}_n$, used to fit the model. This is called the *expected error* of the learning algorithm for this problem (it is no longer a property of a particular dataset). We extend our code to produce a Monte Carlo estimate of this by repeatedly sampling training data to refit the model, then also sampling test data to examine the random design generalisation error.

In the following code we will actually keep track of the inner expectation separately, so we can look at the variability induced by different training sets and see how often one model outperforms, as well as computing the final overall expected error.

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

Even though model 1 is misspecified for this problem, it appears to perform better than model 2 most of the time, despite the function space spanned by model 2 including the true model. We want to try to understand what is happening here, so that we can use it to guide our machine learning modelling.

4.5 Bias-variance decomposition

To do this, we want to decompose the MSE. As an exercise show that:

$$\mathbb{E}_{D_n}\left[\mathbb{E}_{XY}\left[\left(Y-\hat{f}(X)\right)^2\right]\right] = \mathbb{E}_{D_n}\mathbb{E}_{XY}\left[\left(f(X)-\mathbb{E}_{D_n}\hat{f}(X)\right)^2\right] \quad \text{squared bias of model}$$

$$+\,\mathbb{E}_{D_n}\mathbb{E}_{XY}\left[\left(\mathbb{E}_{D_n}\hat{f}(X)-\hat{f}(X)\right)^2\right] = \mathbb{E}_{XY}\mathrm{Var}_{D_n}\hat{f}(X) \quad \text{variance of model fit}$$

$$+\,\mathrm{Var}_{XY}(\varepsilon) \quad \text{irreducible error}$$

Hints:

- Start off by writing Y as $f(X)+\varepsilon$, expand the square and simplify where possible.
- When fully simplified you should have the final variance term above, as well as a term $\mathbb{E}_{D_n}\left[\mathbb{E}_{XY}\left[\left(f(X)-\hat{f}(X)\right)^2\right]\right]$
- Try to further decompose this term by adding and subtracting $\mathbb{E}_{D_n}\hat{f}(X)$ inside the square the term (that is the expected prediction from models over many training samples of size n). Tip: you'll want to gather the four terms into two and expand the square again, simplifying until you reach the final expression above.

Don't worry if you get stuck! We'll prove this in the lectures during the course, but it is good to try yourself before seeing it.

The decomposition above shows that the overall expected mean square error is comprised of 3 key ingredients:

- The bias of the model you are fitting to the problem: in other words, is the model you are fitting able (on average across training sets) to learn the true non-random functional part of the relationship?
- The variance of the model you are fitting to the problem: in other words, how much do the fitted model predictions differ (on average across training sets)?
- Finally the so-called *irreducible error*, which is the random noise in the problem which we could never hope to model.

The beauty of our toy problem is that we can simulate all the terms above. The code gets a bit more complicated, but we can estimate the bias and variance of the two models, and also plot the per-training set biases and variances to see variability.

To do this, we first Monte Carlo estimate $\mathbb{E}_{D_n}\hat{f}(X)$. This means we fit parameters $\hat{\boldsymbol{\beta}}^{(i)}$ for the i-th of N realisations of D_n (ie repeatedly generate training samples of size n). Note that,

$$\begin{aligned}
\mathbb{E}_{D_n}\hat{f}^{(2)}(x) &\approx \frac{1}{N}\sum_{i=1}^{N}\left(\hat{\beta}_0^{(i)} + \sum_{j=1}^{5}\hat{\beta}_j^{(i)}x^j\right) \\
&= \frac{\sum_{i=1}^{N}\hat{\beta}_0^{(i)}}{N} + \sum_{j=1}^{5}\frac{\sum_{i=1}^{N}\hat{\beta}_j^{(i)}}{N}x^j \\
&= \bar{\hat{\beta}}_0 + \sum_{j=1}^{5}\bar{\hat{\beta}}_j x^j
\end{aligned}$$

In other words, the empirical estimate of the term $\mathbb{E}_{D_n}\hat{f}(X)$ simply requires averaging the coefficients of the models fitted to each training sample.

```
>>  Live code online: https://www.louisaslett.com/StatML/notes/  <<
```

These results give a clear picture of what is happening:

- Model 1 is misspecified and therefore suffers bias in the fit, whereas Model 2 contains the truth within its function space and therefore has effectively zero bias (the $\approx 10^{-5} > 0$ value arises as this is a Monte Carlo approximation of the expectation).
- However, model 1 being so simple has low variance, substantially lower than Model 2.
- In particular, note the scales of the bias and variance axes: the reduction in bias achieved by Model 2 is completely swamped by the increase in variance it suffers.
- Overall, Model 1 despite its simplicity and misspecification achieves a lower overall error rate for predictive purposes.

Play around with the above live code boxes. In particular, you can change any of the problem setup variables (denoted by a comment starting `# < =`) in order to vary the experiment. For example, you may want to start by changing model 2's formula to:

```
f2 <- formula(y ~ x + x2) # < = model 2
```

In other words, perfectly specifying the model. You should find the bias remains zero and the variance reduces, giving it superior expected error to model 1, though it still has slightly higher variance (now the differences in bias outweigh those in variance).

In essence, a great deal of machine learning, with its obsessive focus on predictive performance, revolves around finding better ways to optimise this trade off, or find methodologies which tackle one part of the problem head-on (eg variance reduction methods). In the course, we will first cover learning theory (for which this preliminary material provides a good mental picture), followed by methodologies for model fitting. The other key ingredient is practical estimation of the above errors: all the above experiments were only possible because we were simulating from a known truth, a luxury we don't have in reality!

5 Other background

We will assume comfort with linear regression and logistic regression in the course, the APTS Statistical Modelling (Ogden et al., 2021) or any standard undergraduate course being

more than enough background. The initial learning theory we cover (but with an emphasis on machine learning) is intertwined with aspects of decision theory from APTS Statistical Inference (Shaw and Rougier, 2020), though we will cover everything required in a self-contained way in the couse. We will touch on ideas of regularisation in a different setting, but related to, the material in APTS High-dimensional Statistics (Yu, 2021), again that not being pre-requisite as we will cover necessary aspects. Finally, it would be helpful to have a basic knowledge of gradient descent, such as from APTS Statistical Computing (Wilkinson and Wood, 2021) or any standard undergraduate course, and we will make use of simple Monte Carlo and Bootstrap methods, APTS Computer Intensive Statistics (Jenkins et al., 2021) or any standard undergraduate course being more than enough.

References

Bishop, C.M. (2006). Pattern Recognition and Machine Learning, 1st ed, Information Science and Statistics. Springer. URL https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf, ISBN: 978-0387310732

Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science* **16**(3), 199–231. DOI: 10.1214/ss/1009213726

DeGroot, M.H., Schervish, M.J. (2012). Probability and Statistics, 4th ed. Pearson. ISBN: 978-0321500465

Efron, B. (2020). Prediction, estimation, and attribution. *Journal of the American Statistical Association* **115**(530), 636–655. DOI: 10.1080/01621459.2020.1762613

Friedman, J.H. (1998). Data mining and statistics: What's the connection? *Computing Science and Statistics* **29**(1), 3–9.

Hastie, T., Tibshirani, R., Friedman, J. (2009). The Elements of Statistical Learning, 2nd ed, Springer Series in Statistics. Springer. URL https://web.stanford.edu/~hastie/ElemStatLearn/printings/ESLII_print12_toc.pdf, ISBN: 978-0387848570

James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An Introduction to Statistical Learning, 1st ed, Springer Texts in Statistics. Springer. URL https://www.statlearning.com/s/ISLR-Seventh-Printing.pdf, ISBN: 978-1461471370

Jenkins, P.A., Johansen, A.M., Evers, L. (2021). APTS: Computer Intensive Statistics Notes. URL https://warwick.ac.uk/fac/sci/statistics/apts/students/resources/cis-notes.pdf

Jones, M.L. (2018). How we became instrumentalists (again): Data positivism since World War II. *Historical Studies in the Natural Sciences* **48**(5), 673–684. DOI: 10.1525/hsns.2018.48.5.673

Malik, M.M. (2020). A hierarchy of limitations in machine learning. *arXiv* (2002.05193). URL https://arxiv.org/abs/2002.05193

Murphy, K.P. (2012). Machine Learning: A Probabilistic Perspective, 1st ed. MIT Press. ISBN: 978-0262018029

Nagy, G. (1968). State of the art in pattern recognition. *Proceedings of the IEEE* **56**(5), 836–863. DOI: 10.1109/PROC.1968.6414

Ng, A.Y., Jordan, M.I. (2001). On discriminative vs. Generative classifiers: A comparison of logistic regression and naive bayes, in: Proceedings of the 14th International Conference on Neural Information Processing Systems. pp. 841–848. URL http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf

Ogden, H.E., Davison, A.C., Forster, J.J., Woods, D.C., Overstall, A.M. (2021). APTS: Statistical Modelling Notes. URL https://warwick.ac.uk/fac/sci/statistics/apts/students/resources/statmod-notes.pdf

R Core Team (2021). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/

Shaw, S.C., Rougier, J.C. (2020). APTS: Statistical Inference Notes. URL https://warwick.ac.uk/fac/sci/statistics/apts/students/resources/lecturenotes.pdf

Shmueli, G. (2010). To explain or to predict? *Statistical Science* **25**(3), 289–310. DOI: 10.1214/10-STS330

Tukey, J.W. (1962). The future of data analysis. *The Annals of Mathematical Statistics* **33**(1), 1–67. DOI: 10.1214/aoms/1177704711

Vapnik, V.N. (1998). The Nature of Statistical Learning Theory, 2nd ed. Springer. ISBN: 978-0387987804

Wasserman, L.A. (2014). Rise of the machines, in: Lin, X., Genest, C., Banks, D.L., Molenberghs, G., Scott, D.W., Wang, J.-L. (Eds.), Past, Present, and Future of Statistical Science. Chapman; Hall/CRC, pp. 525–536. DOI: 10.1201/b16720

Wilkinson, D.J., Wood, S.N. (2021). APTS: Statistical Computing Notes. URL https://warwick.ac.uk/fac/sci/statistics/apts/students/resources/apts-statcomp.pdf

Yu, Y. (2021). APTS: High-dimensional Statistics Notes. URL https://warwick.ac.uk/fac/sci/statistics/apts/students/resources/hdsnotes.pdf