BIAS AND VARIANCE IN MACHINE LEARNING

A project report submitted to The University of Manchester
for the degree of Bachelors of Science
in the Faculty of Science and Engineering

2023

Luke Braithwaite
10572865
Supervisor: Professor Gavin Brown

Department of Computer Science

Declaration

No portion of the work referred to in this project report has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Abstract

Bias and Variance in Machine Learning

Luke Braithwaite, Supervisor: Professor Gavin Brown

The bias-variance tradeoff is the classical explanation for the generalisation behaviour of machine learning models across supervised learning tasks. This project explores how well the bias-variance framework explains the observed behaviour of machine learning algorithms. We present the bias-variance decomposition for two commonly used regression and classification losses: squared and cross-entropy. These losses are examples of Bregman divergences. We then derive the generalised bias-variance decomposition for Bregman divergences. In addition, we demonstrate that bias-variance theory correctly explains the model behaviour across various supervised learning regimes and, when considered with recent literature, the behaviour of ensemble methods. The report concludes by examining modern deep learning architectures and other large models, demonstrating how classical bias-variance theory must be adapted to explain their behaviour. We explore the double-decent risk curves produced by deep learning architectures. These can be explained by a monotonically decreasing bias and unimodal variance terms. Bias-variance theory remains a valuable framework for reasoning about the behaviour of machine learning models, but there are still open questions. Further research on the dynamics of bias-variance theory and deep learning could help improve practitioners' understanding of deep learning models and lay the groundwork for a unified theory of deep learning.

Acknowledgements

I am extremely grateful to my supervisor, Professor Gavin Brown, for all his support and advice throughout the completion of this project.

Contents

Word Count: 12 118

List of Figures

List of Tables

Notation

Sets and Spaces

$\mathbb{R}^n$	set of $n \times 1$ vectors with real entries.
$\mathbb{R}^n_\Delta$	set of $n \times 1$ probability vectors.
$\mathcal{X}$	input space.
$\mathcal{Y}$	output space.
$\mathcal{H}$	hypothesis space, or the function space of all functions from $\mathcal{X}$ to $\mathcal{Y}$.
$\mathcal{F}$	model family.

Probability Theory

X	random variable.
$P(X)$	probability distribution of the random variable X.
$P(X, Y)$	joint distribution of the random variables X and Y.
$p_X(x)$	probability mass function for the random variable X.
$f_X(x)$	probability density function for the random variable X.
$F_X(x)$	cumulative density function for the random variable X.
$\mathbb{E}[X]$	expected value of the random variable X.

Linear Algebra

$\mathbf{a}$	vector.
$\mathbf{A}$	matrix.
$\mathbf{I}_n$	identity matrix of size n.
$\langle \cdot, \cdot \rangle$	inner product of two vectors.
$\lVert \cdot \rVert_2$	Euclidean distance of a vector or the ℓ^2-norm of a vector.

Machine Learning

$\mathcal{D}, \mathcal{D}_n$	data set of n samples, $D_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$.
$\mathbf{x}$	feature vector.
$\mathbf{y}$	output label or target.
$(\mathbf{x}_i, y_i)$	pair of input vector $\mathbf{x}_i$ with its label y_i.
$f(\,\cdot\,;\mathcal{D})$	model $f \in \mathcal{F}$ trained on the data set $\mathcal{D}$.
$\ell(\mathbf{y}, f(\mathbf{x};\mathcal{D}))$	loss of the model $f(\mathbf{x};\mathcal{D})$.

Acronyms

DNN	deep neural network.
ERM	empirical risk minimiser.
i.i.d.	independent and identically distributed.
LLM	large language model.
ML	machine learning.
SGD	stochastic gradient descent.
SLT	statistical learning theory.

CHAPTER 1

Introduction

Are bigger models always better? Since the rise of deep learning, modern machine learning (ML) has become very big, with companies such as Google and Meta releasing larger and larger models with billions of parameters. These deep/complex neural networks have state-of-the-art performance across various problem domains. Just how large have these models become? If we consider the rise of large language models (LLMs), we can see that over the past five years, the number of model parameters has approximately doubled every three to four months. Each model shown in Figure 1.1 was considered state-of-the-art at the time of release.

To describe the behaviour of these models, we need a formal mathematical framework that allows for reasoning about ML models, preferably in a way that accounts for a model's complexity. This report will use the principle of *empirical risk minimisation* and the larger framework *statistical learning theory*, as this allows for a discussion of model performance in terms of *generalisability* or how well the models can predict and model unseen data. However, we also need an approach applicable to actual models. To do this, we shall explore generalisability through the lens of the *bias-variance tradeoff*, the classical explanation for describing model behaviour and generalisation.

1.1 Project Aims

This project explores the *bias-variance decomposition* of various loss functions and uses these decompositions to examine and explain the model's behaviour. In particular, we would like to know when the bias-variance tradeoff adequately describes a model's behaviour and, importantly, when it does not. With this in mind, we hope to answer the following questions:

1. Is the bias-variance tradeoff a helpful way to describe model behaviour?
2. How can the bias/variance of a model be reduced?

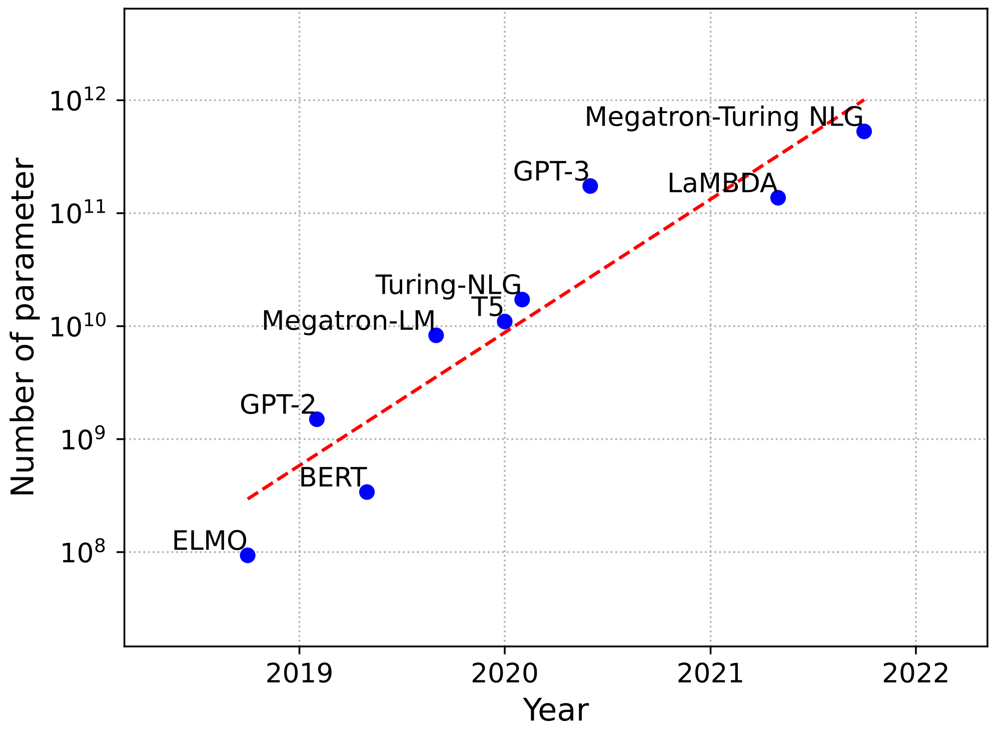

Figure 1.1: Increase in the size of recent large language models; the exponential growth can be seen. Currently, the number of parameters we can train a model on doubles around every four months. The data for this chart is included within Chapter A.

3. When does the classical bias-variance tradeoff break down?

4. Does the bias-variance tradeoff hold for modern ML practice?

1.2 Report Structure

We now give a summary of each chapter.

- Chapter 2 provides the necessary context and background for the bias-variance tradeoff and the fundamental concepts of machine learning. In addition, the bias-variance decomposition for the squared loss is presented.

- Chapter 3 explores the topic of Bregman divergences and derives their generalised bias/variance decomposition. Using this generalised decomposition, we derive the decomposition of the KL divergence, which we use to decompose the cross-entropy loss.

- Chapter 4 looks at the impact of regularisation on the bias and variance of a model.

- Chapter 5 explores ensemble methods and how they can be used to reduce the bias or variance of a model.

- Chapter 6 explores how bias and variance behave for deep learning models and how the classical theory breaks down.

- Chapter 7 summarises and presents the findings of this report.

We will begin our journey into bias and variance by introducing the necessary mathematical background and fundamental machine learning concepts of statistical learning theory and model generalisation.

CHAPTER 2

Setting the Scene

This chapter aims to define the key machine learning concepts and the context needed to understand the later chapters of this report. We shall explore critical machine learning concepts and defined supervised learning tasks; then discuss statistical learning theory and finally move on to define bias and variance in the context of machine learning.

2.1 Supervised Learning

This project focuses on *supervised learning*, where every input has an output 'label' we are trying to predict. Given a data set of independent and identically distributed (i.i.d.) samples drawn from some unknown joint distribution, we try to predict a function f such that $f(\mathbf{x}) = y$. The formal definition is given below as Definition 1 and is adapted from Russell and Norvig [66, p. 671].

Definition 1 (Supervised Learning).
Given a data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=0}^{N}$ drawn i.i.d. from the joint distribution of the vector spaces $\mathcal{X}$ and $\mathcal{Y}$ where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y}$. The aim is to learn a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(\mathbf{x}_i) = \mathbf{y}_i$, a machine learning algorithm $\mathcal{A}$ is used to do this.

2.2 Loss Functions

We generally consider models that are *function approximators*. The aim is to find the function that best approximates the mapping between the input space $\mathcal{X}$ and the output space $\mathcal{Y}$. We shall denote a generic model by $f(\mathbf{x}; \mathbf{w})$ where $\mathbf{x}$ is a feature vector. In most cases, we assume $\mathbf{x} \in \mathbb{R}^d$. The vector or matrix $\mathbf{w}$ are the model parameters, where $\mathbf{w} \in \mathbb{R}^p$ has p parameters.

Loss functions are used to say how good the approximation is; they measure how much the model's output differs from a label for a given input. In regression cases, the output is commonly a single value in $\mathbb{R}$, so the *squared loss* is commonly used,

$$\ell\left(y, f(\mathbf{x}; w)\right) = \frac{1}{2}(y - f(\mathbf{x}; \mathbf{w}))^2. \tag{2.1}$$

As we have a way to measure the performance of the model, we may define *training error*, which is the average loss over a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, or

$$\ell_{\text{train}} = \frac{1}{N}\sum_{i=1}^{N} \ell\left(y_i, f(\mathbf{x}_i; \mathbf{w})\right), \tag{2.2}$$

this converts finding the best function approximator to an optimisation problem. We aim to find $\mathbf{w}^*$ where

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \ell_{\text{train}}.$$

To do this, gradient-based optimisation methods are used.

What about classification tasks? We could use the squared loss and consider classification a discrete regression task. However, this is not ideal; for most classification tasks, we want an idea of the uncertainty of the result, so probabilistic classification is used. To use a classic example, say we want to distinguish between images of cats and dogs. Instead of simply giving the label cat or the label dog, the classifier will return a probability distribution in which each entry is the probability the input is class c. So, the loss function should measure the distance between the distributions. We do not currently have a formal notion of distance, but this will be explored in more detail in Chapter 3. Information-theoretic measures are often used, and cross-entropy is the most common.

Definition 2 (Cross-entropy).
The *cross-entropy* between discrete distributions p and q over the same probability space $\mathcal{X}$, denoted as $H(p, q)$, is defined as follows:

$$H(p, q) \overset{\text{def}}{=} -\sum_{x\in\mathcal{X}} p(x) \log q(x), \tag{2.3}$$

where $p(c)$ is the probability of class c.

Cross-entropy can be considered from the perspective of information theory, which defines cross-entropy as 'the expected number of bits needed to compress some data samples drawn from the distribution p using a distribution code q' [54, p. 205]. It can also be thought of in terms of expectations

$$H(p, q) = -\mathbb{E}_p[\log q]. \tag{2.4}$$

As we have introduced supervised learning and the concept of loss functions, we shall now look at generalisation, overfitting, and underfitting.

2.3 Generalisation

Models are usually trained so *inference* can be performed on unseen data. The performance of a model on unseen data is known as *generalisation*, an unseen validation set commonly used to measure the generalisability of a model. The average over all samples is called the model's *expected risk*.

Before continuing, we define three terms: error, risk, and loss. *Error* is the difference between a single predicted value and a single actual value, *loss* is the average error over the training data set, and *risk* is the average error over all data.

Why do some models generalise well? To answer this, we need to understand the impact of model complexity. If a model is too simple, it will not accurately model the underlying distribution from which the samples are drawn. This means there is a systematic error between the model and the underlying distribution. In this case, the model is *underfitting*. Otherwise, if the model is too complex, it will be too sensitive to small changes in the data set. This is known as *overfitting*. Figure 2.1 demonstrates both overfitting and underfitting in polynomial regression, demonstrating a sweet spot in model complexity.

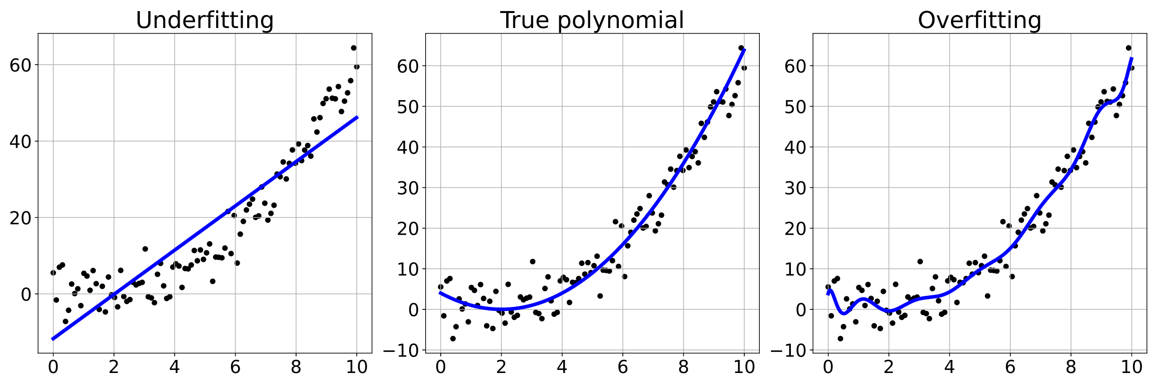

Figure 2.1: Demonstration of underfitting and overfitting when performing polynomial regression where Gaussian noise $\mathcal{N}(0, 5)$ has been added. **(LEFT)** Underfitting. **(CENTRE)** Actual polynomial. **(RIGHT)** Overfitting.

2.4 Empirical Risk Minimisation

statistical learning theory (SLT) is a mathematical framework that can be used to understand and derive learning algorithms. It was developed in the 1960s by the Soviet mathematician Vladamir Vapnik; it was then rediscovered and popularised in the 1990s. One part of this framework most relevant to this project is the Empirical Risk Minimisation principle [75]. In Section 2.2, we stated that we could estimate a model's 'error' by averaging the loss over a data set. SLT calls this the *empirical risk* over a data set $\mathcal{D}_n = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ and is defined as follows.

Definition 3 (Empirical risk of a model).

$$\hat{R}(f, \mathcal{D}_n) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{y}_i, f(\mathbf{x}_i)) \tag{2.5}$$

The *population risk* is the 'true' risk calculated on all possible data we may encounter. Comparing it to the empirical risk, it can be considered the case when $n = \infty$. The population risk includes all data we may encounter, including unseen data. Due to this, it is also known as the *generalisation error*. It is defined below.

Definition 4 (Population risk of a model).

$$R(f) \overset{\text{def}}{=} \mathbb{E}_{(\mathbf{x},\mathbf{y})} \left[\ell(\mathbf{y}, f(\mathbf{x}))\right] = \iint \ell(\mathbf{y}, f(\mathbf{x})) p(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\, \mathbf{x} \, \mathrm{d}\, \mathbf{y}$$

In supervised learning, training data is used to learn the function f. By comparing Equation (2.2) and Definition 3, the training error is simply the empirical risk of the trained model on the training set. So, if we can find the global minima of the training error, the model obtained would minimise the empirical risk. This model is called the empirical risk minimiser (ERM).

Definition 5 (Empirical Risk Minimiser).
The *empirical risk minimiser* is defined as,

$$f_n^* \overset{\text{def}}{=} \arg\min_f \left[\hat{R}(f, \mathcal{D}_n)\right]. \tag{2.6}$$

We can also define the population risk minimiser, the model that minimises the population risk. This is the model that we would like to find. However, it is not computable in real scenarios as it requires the full underlying distribution, which is usually unknown. The definition is similar to Definition 5.

Definition 6 (Population risk minimiser).
The *population risk minimiser*, also known as the *Bayes predictor* or the *Bayes model*, is defined as

$$f^* \overset{\text{def}}{=} \arg\min_f \left[R(f)\right]. \tag{2.7}$$

2.4.1 The Estimation/Approximation Decomposition

So far, the theoretical model can take any form. However, this is rather unrealistic; instead, we shall limit ourselves to the model family $\mathcal{F}$. Consider the model in $\mathcal{F}$ that minimises the population risk. We can define it simply as

$$f^*_{\mathcal{F}} \overset{\text{def}}{=} \arg\min_{f \in \mathcal{F}} R(f). \tag{2.8}$$

The empirical risk is simply an estimate of the population risk; the ERM does not necessarily minimise the population risk. It could be the global minimum for the empirical risk but only a local minimum for the population risk or not a minimum for the population risk at all. There will be a gap between the risk of the Bayes model and the ERM. This difference is known as the *excess risk* or the *generalisation gap*.

Definition 7 (The excess risk of the ERM).
Let f^*_n be the ERM; its excess risk is defined as

$$R_{\text{excess}}(f^*_n) = R(f^*) - R(f^*_n). \tag{2.9}$$

Figure 2.2 visually represents everything we have discussed. The space of all possible functions is called the *hypothesis space* and is denoted $\mathcal{H}$.

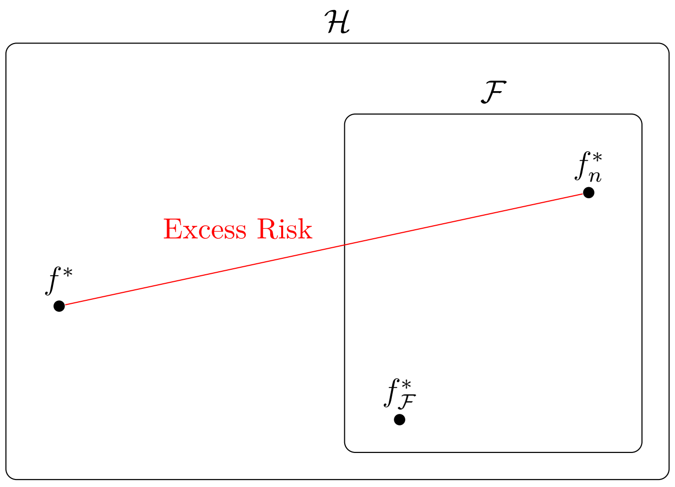

Figure 2.2: Visualisation of the function space we have been discussing where f^*_n is the ERM, $f^*_{\mathcal{F}}$ is the population risk minimiser for a model family $\mathcal{F}$ and f^* is the Bayes model. The excess risk is displayed in red.

This is a good start, but the errors due to the chosen model family and the size of the data set are combined in one term. We want to separate them. Further manipulating Equation (2.9) obtains the *approximation/estimation decomposition*.

Definition 8 (The estimation/approximation decomposition).
Given the ERM f_n^* the *estimation/approximation decomposition* of its excess risk is

$$R_{\text{excess}}(f_n^*) = \underbrace{R(f^*) - R(f_{\mathcal{F}}^*)}_{\text{approximation error}} + \underbrace{R(f_{\mathcal{F}}^*) - R(f_n^*)}_{\text{estimation error}}. \tag{2.10}$$

This decomposition is relatively trivial, but the introduction of the $\pm R(f_{\mathcal{F}}^*)$ terms leads to a helpful interpretation of the two terms.

Approximation error is due to restricting the model family $\mathcal{F}$. In most cases, we have $\mathcal{F} \subset \mathcal{H}$, so most likely $f^* \notin \mathcal{F}$. This depends solely on the chosen model family and can be treated as constant with respect to the random variable $\mathcal{D}_n$.

Estimation error is due to the restriction of the data set size. As we increase the size of the data set, the ERM gets closer to $f_{\mathcal{F}}^*$. This depends on a random variable $\mathcal{D}_n$, so is itself a random variable.

A similar visualisation to Figure 2.2 is possible where the approximation and estimation errors are added.

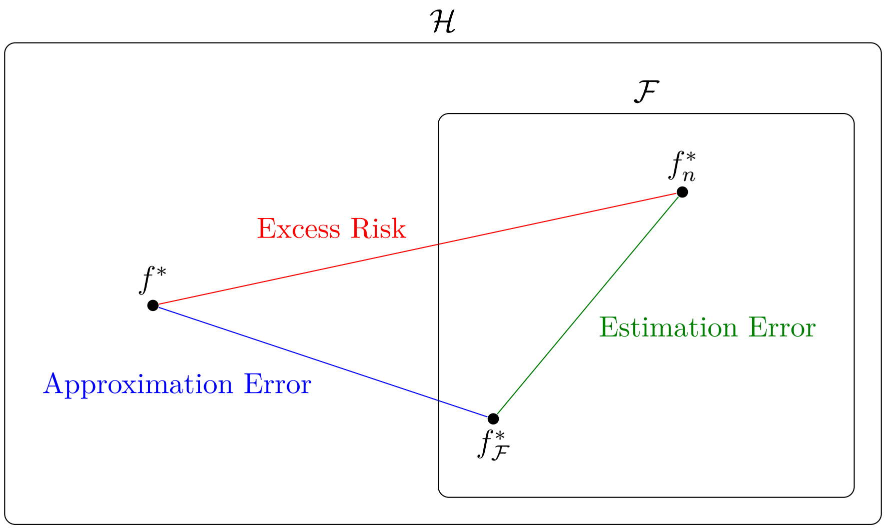

Figure 2.3: Visualisation of the function spaces and models we have discussed above with the added approximation and estimation errors.

Figure 2.3 allows us to relate the approximation/estimation decomposition to model complexity and under/overfitting. We need to get $f_{\mathcal{F}}^*$ as close to f^* to decrease the approximation error. To do this, making $\mathcal{F}$ span more of $\mathcal{H}$ is necessary, which increases model complexity. We shall now deal with the estimation error.

To decrease the estimation error, we could increase the amount of data used to train the model; this results in an ERM closer to $f_{\mathcal{F}}^*$. The larger the model family $\mathcal{F}$, the more training samples are needed to train the model. Another way, given the same data set, is to reduce the model complexity as this will reduce the size of $\mathcal{F}$.

The approximation and estimation errors are in tradeoff, so we choose an appropriate model complexity that gives sufficiently low approximation and estimation errors. Graphing the estimation and approximation error terms as they change with increasing model complexity results in Figure 2.4.

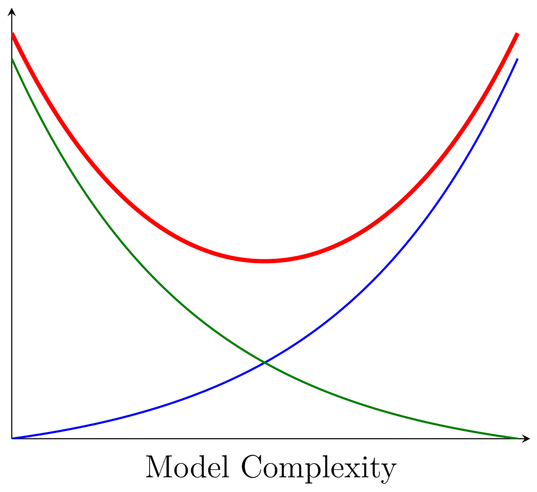

Figure 2.4: An illustration of the approximation error/estimation error tradeoff. The approximation error is shown in green, the estimation error in blue, and the excess risk in red. We can see the signature U-shaped risk curve and the minima or 'sweet spot' that is the tradeoff.

2.4.2 Underfitting and Overfitting

Figure 2.4 describes over/underfitting in terms of the approximation/estimation errors.

Underfitting occurs when the approximation error is too high. Generally, this is when the function space $\mathcal{F}$ is too small relative to the amount of training data.

Overfitting occurs when the estimation error is too high. Generally, this occurs when $\mathcal{F}$ is too large relative to the amount of training data.

2.5 Bias-Variance Decompositions

So far, we have discussed generalisation through the lens of SLT. This approach is rather elegant in principle, but it is very difficult to use in practice. Expect for the most straightforward examples, it is impossible to calculate the Bayes model and the population risk minimiser for a given model class. Instead, the *bias-variance decomposition* is used because it is more convenient to work with and estimate.

Imagine we want to train a model that throws a dart at a dartboard, and we train multiple copies of the same models using different random training sets. We could describe the distribution of the darts in terms of how far they are from the bullseye, how ‘off’ our estimate is from the true value, and how spread or varied the darts are. If the distance from the cluster’s centroid is far from the bullseye, the model has a ‘high bias’, and if the darts are very spread out, the model has a ‘high variance’. Figure 2.5 visualises the effect of bias and variance.

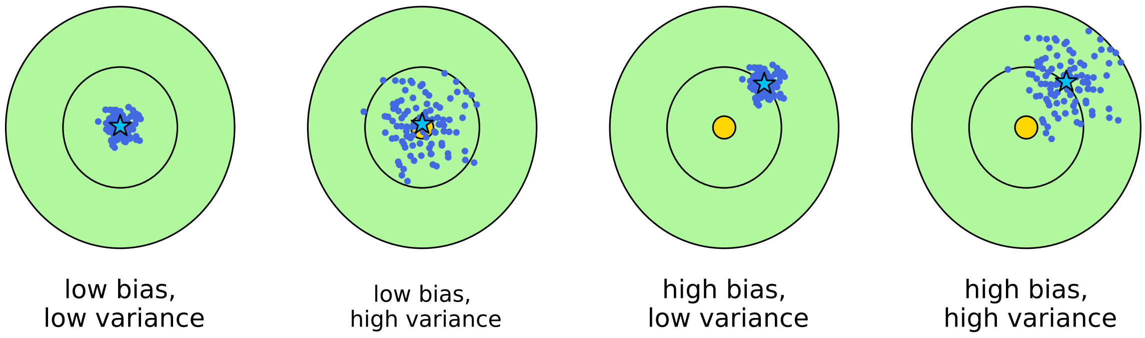

Figure 2.5: The effect of bias and variance. Each dot is a possible model output, the star is the centroid of the predictions, and the bullseye is the true value.

Geman et al. [36] demonstrated that it was possible to decompose the expected squared loss into a bias, variance, and noise term, which we derive in Section 2.6.

2.6 Decomposing the Squared Loss

Before jumping into the derivation, it is helpful to give an overview of what we want to prove and the intermediate results. The first step is to decompose the expected risk of the model into two terms: the noise and the excess risk. The excess risk is then further decomposed into bias and variance terms. If we consider the process like a parse tree, it would look like this.

$$\begin{tikzcd}
 & & \textbf{expected risk} \arrow[dl] \arrow[ddrr] & & \\
 & \textbf{excess risk} \arrow[dl] \arrow[dr] & & & \\
\textbf{bias} & + & \textbf{variance} & + & \textbf{noise}
\end{tikzcd}$$

Figure 2.6: Decomposition tree for the squared loss.

2.6.1 Definitions

We now define both the *expected classifier* and the *Bayes model* for the squared loss. We shall begin with the expected classifier, as this is the simpler of the two; it is the expected value of $f(\mathbf{x}; \mathcal{D})$ over $\mathcal{D}$.

Definition 9 (The expected classifier).
The expected classifier $\bar{f}$ is defined as follows,

$$\bar{f}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] = \int f(\mathbf{x};\mathcal{D}) p\mathcal{D} \, \mathrm{d}\,\mathcal{D}. \tag{2.11}$$

We now move on to the Bayes model, or the population risk minimiser, for the squared loss.

Definition 10 (The Bayes model for the squared loss).
The *Bayes model* for the squared loss is

$$f^*(\mathbf{x}) = \arg\min_{f \in \mathcal{F}} R(f) = \mathbb{E}_{y|\mathbf{x}}[y]. \tag{2.12}$$

Proof. The population risk of squared loss is defined as $R(f) = \mathbb{E}_{(\mathbf{x},y)}[(f(\mathbf{x}) - y)^2]$, the *Bayes model* $f^*(\mathbf{x})$ is the population risk minimiser, so it is simply

$$f^*(\mathbf{x}) = \arg\min_{f} R(f).$$

We begin by expanding the square to obtain

$$f^*(\mathbf{x}) = \mathbb{E}_{(\mathbf{x},y)}[f(\mathbf{x})^2 - 2yf(\mathbf{x}) + y^2].$$

As we want to minimise, we differentiate with respect to $f(\mathbf{x})$. This yields

$$\frac{\partial}{\partial f(\mathbf{x})} \mathbb{E}_{(\mathbf{x},y)}[f(\mathbf{x})^2 - 2yf(\mathbf{x}) + y^2] = \mathbb{E}_{(\mathbf{x},y)}[2f(\mathbf{x}) - 2y]. \tag{2.13}$$

We can then decompose Equation (2.13) into two expectations using the expectation algebra to derive

$$\mathbb{E}_{(\mathbf{x},y)}[2f(\mathbf{x}) - 2y] = \mathbb{E}_{\mathbf{x}}[\mathbb{E}_{y|\mathbf{x}}(2f(\mathbf{x}) - 2y)] = \mathbb{E}_{\mathbf{x}}\left[2f(\mathbf{x}) - 2\,\mathbb{E}_{y|\mathbf{x}}[y]\right]. \tag{2.14}$$

The minimum occurs when Equation (2.14) is 0. Solving this equation gives $f(\mathbf{x}) = \mathbb{E}_{y|\mathbf{x}}[y]$. This gives us $f^*(\mathbf{x}) = \mathbb{E}_{y|\mathbf{x}}[y]$, which is the Bayes model for the squared loss. We can now decompose the expected risk. □

2.6.2 Decomposing the Expected Risk

The first step is to decompose the expected risk into the noise term and the excess risk. This is achieved by introducing an $f^*(\mathbf{x})$ term similar to Definition 8.

Lemma 2.1.
We can decompose the risk of the model into two terms: the noise and the excess risk. Formally, we demonstrate the following equality holds

$$\mathbb{E}_{\mathcal{D}}[R(f)] = \mathbb{E}_{\mathbf{x}}\left[(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x}))^2\right] + \mathbb{E}_{(\mathbf{x},y)}\left[(f^*(\mathbf{x}) - y)^2\right],$$

where the expectations are implicitly conditioned on $\mathcal{D}$.

Proof. As the model is trained using the squared loss, we have

$$\mathbb{E}_{\mathcal{D}}[R(f)] \overset{\text{def}}{=} \mathbb{E}_{(\mathbf{x},y)}\left[(f(\mathbf{x};\mathcal{D}) - y)^2\right].$$

We begin by introducing an $f^*(\mathbf{x})$, which gives us the following:

$$\mathbb{E}_{(\mathbf{x},y)}\left[(f(\mathbf{x};\mathcal{D}) - y)^2\right] = \mathbb{E}_{(\mathbf{x},y)}\left\{([f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x})] + [f^*(\mathbf{x}) - y])^2\right\}.$$

Next, we expand the square and simplify the resulting expression to give us

$$\mathbb{E}_{\mathbf{x}}\left[(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x}))^2\right] + 2\,\mathbb{E}_{(\mathbf{x},y)}[(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x}))(f^*(\mathbf{x}) - y)] + \mathbb{E}_{(\mathbf{x},y)}\left[(f^*(\mathbf{x}) - y)^2\right].$$

Focusing on the middle term, we can push the expectation over y inwards, which leads to

$$\mathbb{E}_{\mathbf{x}}\left[(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x}))\left(f^*(\mathbf{x}) - \mathbb{E}_{y|\mathbf{x}}[y]\right)\right],$$

this gives us $f^*(\mathbf{x}) - \mathbb{E}_{y|\mathbf{x}}[y] = f^*(\mathbf{x}) - f^*(\mathbf{x}) = 0$. Hence we can conclude that

$$\mathbb{E}_{\mathcal{D}}[R(f)] = \underbrace{\mathbb{E}_{\mathbf{x}}\left[(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x}))^2\right]}_{\text{excess risk}} + \underbrace{\mathbb{E}_{(\mathbf{x},y)}\left[(f^*(\mathbf{x}) - y)^2\right]}_{\text{noise}},$$

as desired. □

The second step involves breaking down the excess risk term, which we do by introducing $\bar{f}(\mathbf{x})$ terms.

Lemma 2.2.
The excess risk can be decomposed into bias and variance terms. So, the following equality holds

$$\mathbb{E}_{\mathbf{x}}\left[(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x}))^2\right] = \mathbb{E}_{\mathbf{x}}\left[\left(f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\right)^2\right] + \mathbb{E}_{\mathbf{x}}\left[\left(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\right)^2\right]. \tag{2.15}$$

Proof. First, we introduce $\bar{f}(\mathbf{x})$ terms to give us

$$\mathbb{E}_{\mathbf{x}}\Big[\big(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x})\big)^2\Big] = \mathbb{E}_{\mathbf{x}}\Big\{\Big(\big[f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\big] + \big[\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\big]\Big)^2\Big\}.$$

Next, we expand the square and simplify the resulting expectations; which gives

$$\mathbb{E}_{\mathbf{x}}\Big[\Big(f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\Big)^2\Big] + 2\,\mathbb{E}_{\mathbf{x}}\Big[\Big(f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\Big)\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)\Big] + \mathbb{E}_{\mathbf{x}}\Big[\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)^2\Big].$$

If we focus on the middle term and push the expectation inwards, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{x}}\Big[\Big(f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\Big)\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)\Big] &= \mathbb{E}_{\mathbf{x}}\Big[\Big(\mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - \bar{f}(\mathbf{x})\Big)\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)\Big] \\ &= \mathbb{E}_{\mathbf{x}}\Big[\Big(\bar{f}(\mathbf{x}) - \bar{f}(\mathbf{x})\Big)\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)\Big] \\ &= 0.\end{aligned}$$

This yields the desired equality:

$$\mathbb{E}_{\mathbf{x}}\Big[\big(f(\mathbf{x};\mathcal{D}) - f^*(\mathbf{x})\big)^2\Big] = \underbrace{\mathbb{E}_{\mathbf{x}}\Big[\Big(f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\Big)^2\Big]}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}\Big[\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)^2\Big]}_{\text{bias}}. \qquad \square$$

Theorem 2.3 (The bias-variance decomposition of the squared loss).
The bias-variance decomposition of the squared loss is

$$\mathbb{E}_{\mathcal{D}}[R(f)] = \underbrace{\mathbb{E}_{\mathbf{x}}\Big[\Big(f(\mathbf{x};\mathcal{D}) - \bar{f}(\mathbf{x})\Big)^2\Big]}_{\text{variance}} + \underbrace{\mathbb{E}_{\mathbf{x}}\Big[\Big(\bar{f}(\mathbf{x}) - f^*(\mathbf{x})\Big)^2\Big]}_{\text{bias}} + \underbrace{\mathbb{E}_{(\mathbf{x},y)}\Big[\big(f^*(\mathbf{x}) - y\big)^2\Big]}_{\text{noise}}.$$

Proof. This is a consequence of Lemmas 2.1 and 2.2. $\square$

2.7 Bias, Variance and Noise

Bias measures the mismatch between the model and the underlying data distribution. It is a form of systematic error and occurs when the underlying assumptions of the model do not match the underlying data generation process. In the underfitting example of Figure 2.8, the model has a high bias, as the linear model assumes an underlying linear relationship when it is quadratic.

Variance is the variability in model predictions or the degree to which the model is susceptible to small changes in the training data. In general, it is caused by highly complex models with many parameters; this means the model is more flexible to fit the training data and can learn a more complex mapping. However, instead of fitting the underlying distribution, it might fit to the noise or small changes in the data set. We can connect this to the overfitting example of Figure 2.8 — the model has a high variance. Instead of fitting to the polynomial, it fits to the sample noise.

Noise is a form of irreducible error and is a by-product of the data collection process or the underlying data set. Classic examples of noise are from signal processing, which tends to be due to the sensor's resolution or the sampling rate. However, noise can occur in other data sets. Consider house price data; an example of noise is when you have two identical houses, and one sells for more. In very noisy data, the underlying data generation process can be obscured and is difficult to reduce. We can think of noise as the smallest possible risk achievable.

2.8 Why is the Decomposition Useful?

We have demonstrated that squared loss can be decomposed into bias, variance, and noise terms. You might ask why this is useful. Each quantity can be estimated for a model, showing us the expected risk's sources and causes. Their relationship can then be used to diagnose the cause of a model's generalisation issues. If the variance is the biggest contribution to the expected risk, this suggests the model is overfitting, so either the model complexity should be reduced or regularisation should be added. If bias is the most significant contributor, the model is likely too simple to capture the underlying distribution, so the model complexity should be increased. The same is true for noise, but it suggests a problem with the data and that extra processing may be needed.

2.9 Underfitting and Overfitting: Revisited

Geman et al. [36] demonstrated that the bias and variance decomposition is inextricably linked to the concept of overfitting and underfitting. Overfitting and underfitting are formally defined in terms of bias and variance.

Underfitting occurs when the bias is high because the model has insufficient flexibility to model the underlying distribution properly.

Overfitting occurs when the variance is high, and the model has too much flexibility. Therefore, it models the noise in the data, not the underlying distribution, and is overly sensitive to the training data.

2.10 The Bias-Variance Tradeoff

The bias and variance are in tradeoff; as a model gets more complex, it has more flexibility to model the underlying data generation process, so the bias decreases. However, as model complexity increases, it becomes more sensitive to the initial training data set, so the variance increases. Usually, with increasing model complexity, the bias decreases monotonically, whereas the variance increases monotonically. This gives the characteristic U-shaped risk curve, which is demonstrated in Figure 2.7. The figure also demonstrates why it is called the *bias-variance tradeoff*. It is necessary to add flexibility to the model, which increases the variance of the model to reduce the model bias and minimise the expected risk of the model.

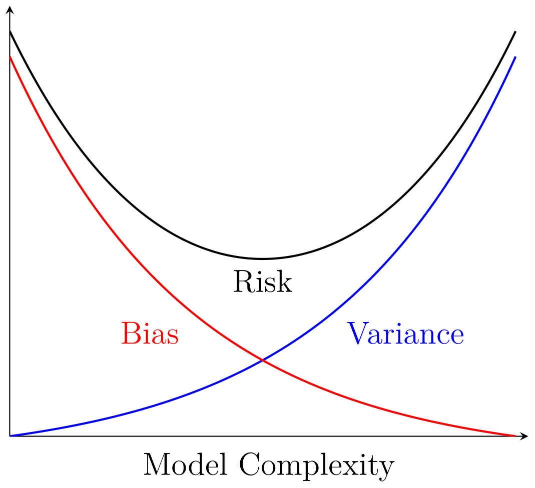

Figure 2.7: This image shows the monotonically decreasing bias, monotonically increasing variance and the U-shaped risk curve. The bias-variance tradeoff aims to find a model complexity that minimises the expected risk

2.11 Revisiting Polynomial Fitting

Figure 2.1 demonstrates that increasing the degree of the polynomial leads to overfitting and decreasing it can lead to underfitting. We can decompose the expected risk of the model to estimate the bias and variance of the model. Figure 2.8 plots all three values against the polynomial's degree, which measures the model complexity. This plot matches our intuitive understanding and the results observed in Figure 2.1.

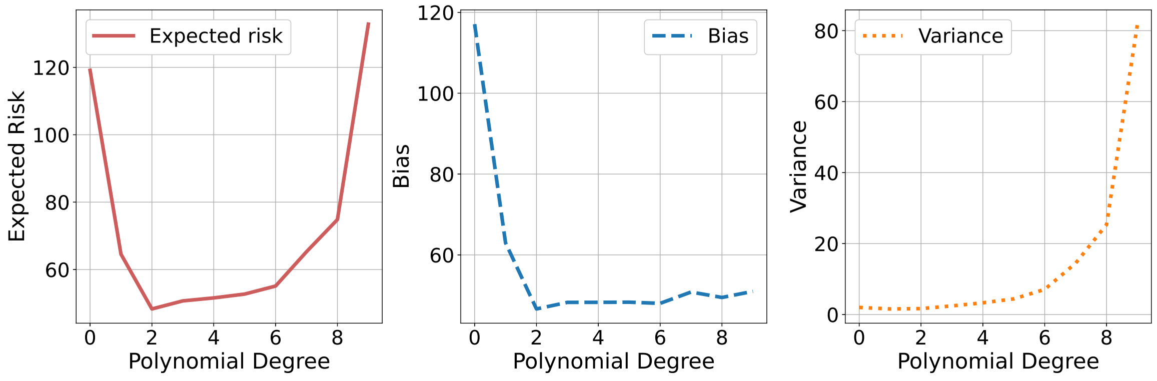

Figure 2.8: The plot of the estimated bias, variance and expected risk for polynomial curve fitting. **(LEFT)** Expected Risk. **(CENTRE)** Bias. **(RIGHT)** Variance.

2.12 Reducing Bias and Variance

How can we reduce the bias or variance of a model? Table 2.1 gives general approaches to reduce either a model's bias or the variance. We shall cover these techniques, give empirical results, and explain how and why these approaches work.

Reduce bias by...	**Reduce variance by...**
• Use a more complex model	• Use a simpler model
• More training samples	• Apply a form of regularisation
• Use an ensemble method (**boosting**)	• Use an ensemble method (**bagging**)

Table 2.1: Approaches to reduce the bias and the variance of a model.

2.13 Estimating Bias and Variance

To estimate the bias and variance of a model, we must approximate the expectation over $\mathcal{D}$. There are two approaches. The first involves partitioning $\mathcal{D}$ into k disjoint subsets, and the second uses bootstrap sampling.

2.13.1 Data Set Partitioning

If $\mathcal{D}$ is partitioned into k disjoint subsets such that $\mathcal{D} = \bigcup_{i=1}^{k} \mathcal{D}^{(i)}$, the bias and variance of $f_{\mathcal{D}^{(i)}}$ can be calculated by taking the expectation over $\mathcal{D}^{(i)}$ estimates the bias and variance of the model. Algorithm 1 describes this approach in more detail.

Algorithm 1 data set partitioning approach

```
Partition $\mathcal{D}$ into $\mathcal{D}^1, \dots, \mathcal{D}^k$
for $i = 1, \dots, k$ do
    Train $f_{\mathcal{D}^{(i)}}$ using $\mathcal{D}^{(i)}$
    Calculate $\hat{y}_i$ using evaluation set
Calculate expected risk, bias and variance from $\hat{y}$
return expected risk, bias, variance
```

2.13.2 Bootstrap Sampling

The second approach involves taking repeated bootstrap samples (sampling with replacement) and then taking the expectation over them to estimate the bias and variance of the model. Algorithm 2 describes this approach in more detail.

Algorithm 2 Bootrap approach

```
for $i = 1, \dots, k$ do
    $\mathcal{B} \leftarrow$ BOOSTRAP($\mathcal{D}$)
    Train $f_{\mathcal{B}}$ using the bootstrap sample $\mathcal{B}$
    Calculate $\hat{y}_i$ using evaluation set
Calculate expected risk, bias and variance from $\hat{y}$
return expected risk, bias, variance
```

2.14 Chapter Summary

This chapter briefly introduced supervised learning and defined the notion of a loss function and training loss. We then introduced SLT and the empirical risk minimisation principle, which gives us a formal mathematical framework to understand the behaviour of ML algorithms. Next, we defined a model's bias, variance, and noise and presented the bias-variance decomposition for the squared loss. This is the classical theory to explain the model generalisation and behaviour. We concluded by discussing how we can estimate the bias and variance of a model and techniques to reduce a model's bias or variance. In the next chapter, we generalise our notion of loss functions and distances by introducing Bregman divergences and their generalised bias-variance decomposition.

CHAPTER 3

Bregman Divergences

Chapter 2 discussed how loss functions can be seen as a measure of the distance between the predicted and actual output. This chapter uses *Bregman divergences* [13] to define distance formally. This leads to the generalised bias-variance decomposition for Bregman divergences; this will be used to derive the decomposition of the cross-entropy loss. We shall begin by exploring the squared loss.

3.1 Geometry of the Squared Loss

The following section introduces Bregman divergences by exploring a geometric interpretation of the squared loss and has been adapted from Ried [65]. In Section 2.6, we derived the bias-variance decomposition for the squared loss. The 1D squared loss can be generalised to $\mathbb{R}^n$. It is the squared Euclidean distance between two vectors: the prediction and the label. If we consider the geometry of the squared loss, something interesting happens. Let $\langle \mathbf{x}, \mathbf{y} \rangle = \sum_i x_i y_i$ be the inner product. The squared loss can be expressed as

$$\|\mathbf{x} - \mathbf{y}\|^2 = \langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle = \|\mathbf{x}\|^2 - \|\mathbf{y}\|^2 - \langle 2\,\mathbf{y}, \mathbf{x} - \mathbf{y} \rangle,$$

by the linearity of the inner product. This formulation of the squared loss lends to a nice geometric interpretation; $2\,\mathbf{y}$ is the derivative of $\|\mathbf{y}\|^2$, and $\|\mathbf{y}\|^2 + \langle 2\,\mathbf{y}, \mathbf{x} - \mathbf{y} \rangle$ is the value of the line tangent to $\|\mathbf{y}\|^2$ evaluated at $\mathbf{x}$. So, the squared loss is the difference between the function $f(\mathbf{x}) = \|\mathbf{x}\|^2$ and the tangent at $\mathbf{y}$ evaluated at $\mathbf{x}$. More formally

$$\|\mathbf{x} - \mathbf{y}\|^2 = f(\mathbf{x}) - \underbrace{\left(f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle\right)}_{\text{tangent of } f \text{ at } \mathbf{y} \text{ evaluated at } \mathbf{x}}. \tag{3.1}$$

Figure 3.1 visualises the 1D case.

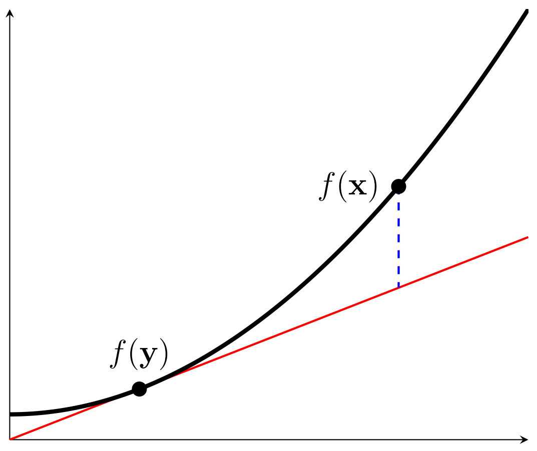

Figure 3.1: A visualisation of the squared loss as a Bregman divergence. The curve $\|\mathbf{x}\|^2$ is plotted in black, the line tangent to $f(\mathbf{y})$ is plotted in red, and the squared Euclidean distance is the length of the line plotted in blue.

The squared loss must always be positive, a negative distance would not be sensible, so by Equation (3.1), we have

$$f(\mathbf{x}) \geqslant f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n,$$

which Boyd and Vandenberghe [12, p. 69] states is equivalent to saying f is a convex function given it is sufficiently differentiable. This means we can derive a distance D_f with similar structure and properties to the squared loss by choosing a convex function f and defining

$$D_f(\mathbf{x}, \mathbf{y}) \stackrel{\text{def}}{=} f(\mathbf{x}) - (f(\mathbf{y}) + \langle \nabla f(\mathbf{y}), \mathbf{x} - \mathbf{y} \rangle).$$

Any distance we define this way is a Bregman divergence, and the convexity of f guarantees nonnegativity for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$.

3.2 Bregman Divergences

We have motivated the study of Bregman divergences by exploring the squared loss. We now give the formal definition.

Definition 11 (Bregman Divergence).
Let $F : \mathcal{S} \to \mathbb{R}$ be a strictly convex differentiable function, then the *Bregman Divergence* derived from F is a function $D_F : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_+$ such that

$$D_F(x, y) \stackrel{\text{def}}{=} F(x) - (F(y) + \langle \nabla F(y), x - y \rangle)$$

Table 3.1 gives examples of common loss functions that are also Bregman divergences.

Loss function	**Bregman Divergence** $D_F(p,q)$	**Generator** $F(p)$	**Domain** $\mathcal{S}$
Squared loss	$\lVert \mathbf{p}-\mathbf{q} \rVert^2$	$\lVert \mathbf{q} \rVert^2$	$\mathbf{p} \in \mathbb{R}^n$
Poisson loss	$\frac{p}{q} - \ln \frac{p}{q} - 1$	$-\ln p$	$p \in \mathbb{R}_+$
KL Divergence	$\sum_i \mathbf{p}_i \log \frac{\mathbf{p}_i}{\mathbf{q}_i} + \sum_i \mathbf{p}_i + \sum_i \mathbf{q}_i$	$\sum_i \mathbf{p}_i \log \mathbf{p}_i$	$\mathbf{p} \in \mathbb{R}^n_\Delta$
Itakuru-Saito	$\sum_i \left(\frac{\mathbf{p}_i}{\mathbf{q}_i} - \log \frac{\mathbf{p}_i}{\mathbf{q}_i} - 1 \right)$	$-\sum_i \log \mathbf{p}_i$	$\mathbf{p} \in \mathbb{R}^n_+$

Table 3.1: Examples of common loss functions as Bregman divergences. Source: [80]

3.3 The Generalised Bias-Variance Decomposition of Bregman Divergences

Pfau [59] derived the generalised bias-variance decomposition for Bregman divergences; this greatly simplifies decomposing the cross-entropy loss. We present the proof and use it to decompose the squared loss and KL divergence.

Lemma 3.1 (Minimum Expected Bregman Divergence).
Let $F : \mathcal{S} \to \mathbb{R}$ be a strictly convex differentiable function and X be a random variable on $\mathcal{S}$. Then we have

(1) $x^* = \arg\min_z D_F[z \parallel X]$ if and only if $\nabla F(x^*) = \mathbb{E}[\nabla F(X)]$

(2) $\mathbb{E}[X] = \arg\min_z D_F[X \parallel z]$.

Proof. For x^* to minimise the expected divergence, its gradient must be zero. The gradient of the expected Bregman divergence taken over the second argument is given by

$$\begin{aligned}\nabla_z \mathbb{E}\left[D_F[z \parallel X]\right] &= \nabla_z \mathbb{E}\left[F(z) - (F(X) + \langle \nabla F(X), z - X\rangle)\right] \\ &= \nabla_z F(z) - \nabla_z \mathbb{E}\left[\langle \nabla F(X), z - X\rangle\right] \\ &= \nabla_z F(z) - \nabla_z \left[\langle \mathbb{E}[\nabla F(X)], z\rangle\right] \\ &= \nabla_z F(z) - \nabla \mathbb{E}[F(X)] = 0.\end{aligned}$$

The final step allows us to conclude $\nabla F(z) = \nabla \mathbb{E}[F(X)]$. The intermediate steps are the linearity of expectations and the independence of X and z. As F is strictly convex, x^* is a unique global minimum if it exists that satisfies this condition.
We now consider taking the expectation over the first argument. The gradient is

$$\begin{aligned}\nabla_z \mathbb{E}\left[D_F[X \parallel z]\right] &= \nabla_z \mathbb{E}\left[F(X) - (F(z) + \langle \nabla F(z), X - z\rangle)\right] \\ &= -\nabla F(z) - \nabla^2 F(z)\,\mathbb{E}[X] + \nabla^2 F(z) z + \nabla F(z) \\ &= -\nabla^2 F(z)\,\mathbb{E}[X] + \nabla^2 F(z) z = 0.\end{aligned}$$

The final step yields $\nabla^2 F(z)\,\mathbb{E}[X] = \nabla^2 F(z) z$, as F is strictly convex, its Hessian is positive definite and invertible by the invertible matrix theorem [see 78]. We can conclude that $\mathbb{E}[X] = z$. □

Lemma 3.1 restates a key result of Bregman divergences: given a random vector, the expected value minimises the Bregman divergence. This was first proven by Banerjee et al. [4]. From Lemma 3.1, we can decompose the expected Bregman divergence over a random variable for each element $s \in \mathcal{S}$. This results in Theorem 3.2.

Theorem 3.2 (Decomposition of Expected Bregman Divergence).
Let $F : \mathcal{S} \to \mathbb{R}$ be a strictly convex differentiable function and X be a random variable on $\mathcal{S}$. For a value $s \in \mathcal{S}$, the expected Bregman divergences have the following exact decompositions.

(1) $\mathbb{E}\left[D_F[s \parallel X]\right] = D_F[s \parallel x^*] + \mathbb{E}\left[D_F[x^* \parallel X]\right]$ where $x^* = \arg\min_z D_F[z \parallel X]$

(2) $\mathbb{E}\left[D_F[X \parallel s]\right] = D_F[x^* \parallel s] + \mathbb{E}\left[D_F[X \parallel x^*]\right]$ where $x^* = \arg\min_z D_F\left[X \parallel z\right] = \mathbb{E}[X]$

Proof. We begin with the expectation over the second argument. Applying Definition 11 gives

$$\begin{aligned} D_F[s \parallel x^*] + \mathbb{E}\left[D_F[x^* \parallel X]\right] = F(s) - F(x^*) + \langle \nabla F(x^*), s - x^* \rangle \\ + \mathbb{E}\left[F(x^*) - F(X) + \langle \nabla F(X), x^* - X \rangle\right]. \end{aligned}$$

As x^* is a constant, it can be extracted from the expectation and cancelled with $-F(x^*)$. Applying Lemma 3.1 and moving the constants inside the expectation yields

$$\mathbb{E}\left[F(s) + \langle \nabla F(X), s - x^* \rangle - F(X) + \langle \nabla F(X), x^* - X \rangle\right].$$

As inner products are linear,

$$\mathbb{E}\left[F(s) - F(X) + \langle \nabla F(X), s - x^* + x^* - X \rangle\right] = \mathbb{E}\left[F(s) - F(X) + \langle \nabla F(X), s - X \rangle\right],$$

which is $\mathbb{E}\left[D_F[s \parallel X]\right]$ as required.
We now consider the expectation over the first argument. Substituting in $x^* = \mathbb{E}[X]$ and apply Definition 11 gives

$$\begin{aligned} D_F[\mathbb{E}[X] \parallel s] + \mathbb{E}\left[D_F[X \parallel \mathbb{E}[X]]\right] = F(\mathbb{E}[X]) - (F(s) + \langle \nabla F(s), \mathbb{E}[X] - s \rangle) \\ + \mathbb{E}\left[F(X) - (F(\mathbb{E}[X]) + \langle \nabla F(\mathbb{E}[X]), X - \mathbb{E}[X] \rangle)\right]. \end{aligned}$$

Expectations are linear, so the expression can be further decomposed to

$$-(F(s) + \langle \nabla F(s), \mathbb{E}[X] - s \rangle) + \mathbb{E}\left[F(X)\right] - \langle \nabla F(\mathbb{E}[X]), \mathbb{E}[X] - \mathbb{E}[X] \rangle.$$

If an inner product has the form $\langle \cdot, 0\rangle$ or $\langle 0, \cdot\rangle$, then its value is also zero. Hence, $\langle \nabla F(\mathbb{E}[X]), \mathbb{E}[X] - \mathbb{E}[X]\rangle$ is zero. Moving the expectation to the outside and taking advantage of the independence of X and s gives

$$\mathbb{E}\left[F(X) - (F(s) + \langle \nabla F(s), X - s\rangle)\right],$$

which is $\mathbb{E}\left[D_F(X \parallel s)\right]$ as required. □

Suppose that we want to predict a random variable $Y \in \mathcal{S}$ dependent on another variable $X \in \mathcal{R}$. We have the training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ of data samples drawn i.i.d. from the joint distribution, so the task is to learn the function $f_\mathcal{D} : \mathcal{R} \to \mathcal{S}$. If the loss function of the model is the Bregman divergence defined from F, the expected loss can be decomposed exactly using Theorem 3.3.

Theorem 3.3 (Generalised Bias-Variance Decomposition).
Let $F : \mathcal{S} \to \mathbb{R}$ be a strictly convex differentiable function, $f_\mathcal{D} : \mathcal{R} \to \mathcal{S}$ be the model trained on $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, and Y be the random variable we predict from X. The expected Bregman divergence of the data obeys the general bias-variance decomposition:

$$\begin{aligned}&\mathbb{E}_{\mathcal{D},Y}[D_F[Y \parallel f_\mathcal{D}(X)]]\\&\quad= \underbrace{\mathbb{E}_Y[D_F[Y \parallel f^*(X)]]}_{\text{Noise}} + \underbrace{D_F[f^*(X) \parallel \bar{f}(X)]}_{\text{Bias}} + \underbrace{\mathbb{E}_\mathcal{D}[D_F[\bar{f}(X) \parallel f_\mathcal{D}(X)]]}_{\text{Variance}}\end{aligned}$$

where $f^*(X) = \arg\min_z \mathbb{E}_Y[D_F[Y \parallel z]] = \mathbb{E}_Y[Y]$, $\bar{f}(X) = \arg\min_z \mathbb{E}_\mathcal{D}[D_F[z \parallel f_\mathcal{D}(X)]]$ and all expectations are implicitly conditioned on X.

Proof. This is an application of Theorem 3.2. The expectation $\mathbb{E}_{\mathcal{D},Y}[D_F[Y \parallel f_\mathcal{D}(X)]]$ can be written as $\mathbb{E}_\mathcal{D}[\mathbb{E}_Y[D_F[Y \parallel f_\mathcal{D}(X)]]]$. Applying Theorem 3.2 (1) and simplifying the resulting expectation gives

$$\mathbb{E}_\mathcal{D}[\mathbb{E}_Y[D_F[Y \parallel f^*(X)]] + D_F[f^*(X) \parallel f_\mathcal{D}(X)]].$$

The right-hand term can be further reduced by using Theorem 3.2 (2) which yields

$$\mathbb{E}_Y[D_F[Y \parallel f^*(X)]] + \mathbb{E}_\mathcal{D}[D_F[f^*(X) \parallel \bar{f}(X)] + D_F[\bar{f}(X) \parallel f_\mathcal{D}(X)]]].$$

Simplifying the expectation by extracting constants yields

$$\underbrace{\mathbb{E}_Y[D_F[Y \parallel f^*(X)]]}_{Noise} + \underbrace{D_F[f^*(X) \parallel \bar{f}(X)]}_{Bias} + \underbrace{\mathbb{E}_\mathcal{D}[D_F[\bar{f}(X) \parallel f_\mathcal{D}(X)]]}_{Variance},$$

which is the desired decomposition. □

Theorem 3.3 demonstrates the power of Bregman divergences. As long as we formulate a loss function to be a Bregman divergence, we know its exact bias-variance decomposition. To give a concrete example, we again look at the squared loss.

Example (Squared loss decomposition). In Section 2.6, we gave a possible derivation of the bias-variance decomposition of the squared loss. Section 3.1 demonstrated that the squared loss is also a Bregman divergence, so we can restate Theorem 2.3 using Theorem 3.3. We shall begin with $f^*(\mathbf{x})$, which is $\mathbb{E}_{y|\mathbf{x}}[Y]$, which in turn is just Definition 10. Next is $\bar{f}(\mathbf{x})$, by Lemma 3.1 we have $\nabla F(\bar{f}(\mathbf{x})) = \mathbb{E}_{\mathcal{D}}\left[\nabla F(f_{\mathcal{D}})\right]$, which implies that $\bar{f} = \mathbb{E}_{\mathcal{D}}[f_{\mathcal{D}}]$. This matches Theorem 2.3.

3.4 The KL Divergence

In classification, it is common to have a probability distribution over all possible classes in the form of a probability vector. This means the loss function measures the 'distance' between the true probability distribution and the predicted distribution. Chapter 2 introduced the cross entropy loss function, which is commonly used for classification problems and can be considered a measure of the distance between two distributions. It would be nice if cross-entropy is a Bregman divergence. Unfortunately, it is not. Instead, we exploit its connection to *KL divergence* [48].

3.4.1 Motivating the KL Divergence

Consider the probability distributions of two coins C_1 and C_2 that follow the probability distributions given in Table 3.2.

$\Pr(C_i = c)$	H	T
C_1	p_1	p_2
C_2	q_1	q_2

Table 3.2: Distribution of C_1 and C_2

Say we want to measure the distance between the probability distribution of two coins. One way is to measure how hard it is to distinguish the distribution, or for a given sequence of events $\mathcal{S}$ drawn from C_1, how similar is the likelihood given each distribution. This can be expressed as

$$\text{distance} = \frac{\Pr(\mathcal{S}|C_1)}{\Pr(\mathcal{S}|C_2)}.$$

Given a sequence of N independent coin flips that contain N_H heads and N_T tails, the log distance (normalised to the sample size) is

$$\log\left(\frac{p_1^{N_H/N} p_2^{N_T/N}}{q_1^{N_H/N} q_2^{N_T/N}}\right) = \log\left(\frac{p_1^{N_H} p_2^{N_T}}{q_1^{N_H} q_2^{N_T}}\right)^{\frac{1}{N}}.$$

Applying basic log rules yields

$$\frac{N_H}{N}\log\frac{p_1}{q_1}+\frac{N_T}{N}\log\frac{p_2}{q_2}. \tag{3.2}$$

As $\mathcal{S}$ is drawn from C_1, $\frac{N_H}{N}\to p_1$ and $\frac{N_T}{N}\to p_2$ as $N\to\infty$. This means Equation (3.2) can be expressed as

$$p_1\log\frac{p_1}{q_1}+p_2\log\frac{p_2}{q_2}. \tag{3.3}$$

Equation (3.3) defines the KL divergence between C_1 and C_2. This is a simple example but can be generalised to any probability distribution.

3.4.2 The KL Divergence and Negative Entropy

Definition 12 (The Discrete KL Divergence).
Let p and q be discrete probability distributions over a probability space $\mathcal{X}$, the *Kullback-Leibler (KL) divergence* denoted $D_{KL}(p \parallel q)$ is defined as

$$D_{KL}(p \parallel q) \overset{\text{def}}{=} \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)}.$$

Like cross-entropy, it can be expressed as an expectation. It is the expectation over p of the difference of the logs of the distributions or

$$\mathbb{E}_p[\log p - \log q].$$

This is similar to cross-entropy. Given a set of observations drawn from p, the cross entropy is the reciprocal of the likelihood of observing the same observations given by q. This suggests a connection between the two quantities that will be formally defined later in this chapter.

Table 3.1 stated that the KL divergence was the Bregman divergence generated by negative entropy. We now prove this statement.

Theorem 3.4 (KL Divergence and Negative Entropy).
The KL divergence is a Bregman divergence where F is the negative entropy of the distribution or

$$F(p)=\sum_{x\in\mathcal{X}} p(x)\log p(x).$$

PROOF. We begin by finding $\nabla F(p)$, applying the chain rule we get

$$\nabla F(p)=\sum_{x\in\mathcal{X}}\log p(x)+1.$$

Then we have

$$\begin{aligned}
D_F[p \parallel q] &= F(p) - (F(q) + \langle \nabla F(q), p - q \rangle) \\
&= \sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}} q(x)\log q(x) - \left\langle \sum_{x\in\mathcal{X}} \log q(x) + 1, p - q \right\rangle \\
&= \sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}} q(x)\log q(x) - \sum_{x\in\mathcal{X}} p(x)\log q(x) \\
&\quad - \sum_{x\in\mathcal{X}} p(x) + \sum_{x\in\mathcal{X}} q(x)\log q(x) + \sum_{x\in\mathcal{X}} q(x) \\
&= \sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}} p(x)\log q(x) - \sum_{x\in\mathcal{X}} p(x) + \sum_{x\in\mathcal{X}} q(x) \qquad (*)
\end{aligned}$$

Equation ($*$) is the *generalised KL divergence*, but as p and q are probability distributions, they must sum to one. This gives us

$$\sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}} p(x)\log q(x) = \sum_{x\in\mathcal{X}} p(x)\log\frac{p(x)}{q(x)} = D_{KL}(p \parallel q). \qquad \square$$

Heskes [39] first derived the bias-variance decomposition for the KL divergence; however, this proof is somewhat unintuitive and involved. Instead, we will use the generalised bias-variance decomposition for Bregman divergences.

3.5 Decomposing the KL Divergence

Consider a K-class classification task with the training set $\{(\mathbf{x}_i, c_i)\}_{i=1}^N$ drawn i.i.d. from the joint distribution of the random variables X and Y, where $\mathbf{x}_i \in \mathbb{R}^d$. Let $\pi_0 \in \mathbb{R}^K_\Delta$ be the one-hot encoding of the ground truth labels and $\pi(\mathbf{x};\mathcal{D}) \in \mathbb{R}^K_\Delta$, denoted π, the classifier's prediction and is a random variable dependent on X and $\mathcal{D}$. If the model's loss function is $D_{KL}(\pi_0 \parallel \pi)$, then the expected loss can be exactly decomposed using Theorem 3.3.

We first define both $f^*(X)$ and $\bar{f}(X)$ as required by Theorem 3.3. Beginning with $f^*(X)$, denoted π^*, which is simply $\mathbb{E}_Y[\pi_0]$. Next, $\bar{f}(X)$ or the *expected classifier* denoted $\bar{\pi}$.

Lemma 3.5 (The Expected Classifier).
Given a probability vector π, the expected classifier $\bar{\pi}$ is the probability vector such that $\bar{\pi} = \arg\min_z \mathbb{E}[z \parallel \pi]$ and the vector has entries

$$\bar{\pi}[c] = \frac{\exp\mathbb{E}[\ln\pi[c]]}{\sum_{c=1}^K \exp\mathbb{E}[\ln\pi[c]]},$$

where all expectations are implicitly over $\mathcal{D}$ and X.

PROOF. By Lemma 3.1, $\nabla F(\bar{\pi}) = \mathbb{E}[\nabla F(\pi)]$. Differentiating F (using the natural logarithm) and tidying up the resulting summation gives

$$\sum_{c=1}^{K} \ln \bar{\pi}[c] = \sum_{c=1}^{K} \mathbb{E}[\ln \pi[c]], \tag{3.4}$$

where $\ln \bar{\pi}[c] \propto \mathbb{E}[\ln \pi[c]]$ or $\bar{\pi}[c] \propto \exp \mathbb{E}[\ln \pi[c]]$. As $\bar{\pi}$ is a probability distribution, we must normalise it so that it sums to one; this can be done by dividing each entry by the sum of the vector entries. This gives

$$\bar{\pi}[c] = \frac{\exp \mathbb{E}[\ln \pi[c]]}{\sum_{c=1}^{K} \exp \mathbb{E}[\ln \pi[c]]}$$

as required and completes the proof. □

We can now derive the exact decomposition of the KL divergence.

Theorem 3.6 (The Bias-Variance Decomposition of the KL Divergence).
Let $\pi_0 \in \mathbb{R}_{\Delta}^{K}$ is the one-hot encoded ground truth and $\pi \in \mathbb{R}_{\Delta}^{K}$ is the predicted distribution and a random variable dependent on X and $\mathcal{D}$. The bias-variance decomposition of the KL divergence is

$$\mathbb{E}[D_{KL}(\pi_0 \parallel \pi)] = \underbrace{\mathbb{E}_Y[D_{KL}[\pi_0 \parallel \pi^*]]}_{Noise} + \underbrace{D_{KL}[\pi^* \parallel \bar{\pi}]}_{Bias} + \underbrace{\mathbb{E}_{\mathcal{D}}[D_{KL}[\bar{\pi} \parallel \pi]]}_{Variance},$$

where $\pi^* = \mathbb{E}_Y[\pi_0]$ and $\bar{\pi}$ is the expected classifier.

PROOF. This is a consequence of Theorem 3.3 and Lemma 3.5. □

3.6 The Cross-Entropy Connection

Chapter 2 introduced cross-entropy and gave its definition. Manipulating the definition gives

$$\begin{aligned}
H(p,q) &\overset{\text{def}}{=} -\sum_{x\in\mathcal{X}} p(x)\log q(x) \\
&= -\sum_{x\in\mathcal{X}} p(x)\log q(x) + \sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}} p(x)\log p(x) \\
&= \underbrace{\sum_{x\in\mathcal{X}} p(x)\log p(x) - \sum_{x\in\mathcal{X}} p(x)\log q(x)}_{D_{KL}(p\|q)} \quad \underbrace{-\sum_{x\in\mathcal{X}} p(x)\log p(x)}_{H(p)} \\
&= D_{KL}(p \parallel q) + H(p).
\end{aligned} \tag{3.5}$$

So, the cross entropy of p and q can be expressed as the sum KL divergence of p and p and another term, called the *entropy* of p and is denoted $H(p)$. We can now decompose the cross-entropy loss.

Theorem 3.7 (The Bias-Variance Decomposition of the Cross-Entropy Loss).
The bias-variance decomposition for cross-entropy loss is

$$\mathbb{E}[H(\pi_0, \pi)] = \underbrace{\mathbb{E}_Y[D_{KL}[\pi_0 \parallel \pi^*]]}_{Noise} + \underbrace{D_{KL}[\pi^* \parallel \bar{\pi}]}_{Bias} + \underbrace{\mathbb{E}_{\mathcal{D}}[D_{KL}[\bar{\pi} \parallel \pi]]}_{Variance},$$

where $\pi^* = \mathbb{E}_Y[\pi_0]$ and $\bar{\pi}$ is the expected classifier.

Proof. Equation (3.5) and the expectation algebra rules gives

$$\mathbb{E}[H(\pi_0, \pi)] = \mathbb{E}[H(\pi_0) + D_{KL}(\pi_0 \parallel \pi)] = \mathbb{E}[H(\pi_0)] + \mathbb{E}[D_{KL}(\pi_0 \parallel \pi)].$$

Consider $\mathbb{E}[H(\pi_0)]$, as π_0 is a one-hot vector, so its elements are either 1 or 0. This means $H(\pi_0)$ and $\mathbb{E}[H(\pi_0)]$ will be 0, so $\mathbb{E}[H(\pi_0, \pi)] = \mathbb{E}[D_{KL}(\pi_0 \parallel \pi)]$. Applying Theorem 3.6 gives the desired decomposition and completes the proof. $\square$

3.7 Chapter Summary

This chapter introduced Bregman divergences. We began by demonstrating that the squared loss was a Bregman divergence. Next, we presented generalised bias-variance decomposition for Bregman divergence and demonstrated that the bias-variance decomposition of the squared loss was just a special case. Then we moved on to the KL-Divergence and exploited its connection to cross-entropy to decompose the cross-entropy loss. We now have the decomposition for both the squared and cross-entropy losses. The following two chapters use these decompositions to explore the behaviour of two standard machine learning techniques: regularisation and ensemble methods. We begin with regularisation in the next chapter.

CHAPTER 4

Regularisation

For models to generalise well, we need to prevent overfitting. This is especially important as current machine learning practice focuses on large complex models. We want the simplest possible model with sufficient generalisation performance to do this. Or, given a set of models with the same expected risk, we want the simplest. This is achieved by penalising overly complex models and is known as *regularisation*.

Regularisation approaches can be classified in many ways, but for the sake of this discussion, we will group regularisation approaches into two key types:

- **Explicit regularisation** is when a regularisation term is explicitly added to the objective function, usually as a penalty term.
- **Implicit regularisation** is all other forms of regularisation and tends to be more common in deep learning. Examples of this are early stopping [61, 82] during training if we detect the model starting to overfit and gradient-based optimisation method such as stochastic gradient descent (SGD)[1] [6, 23, 30, 64, 71].

This chapter focuses on explicit regularisation and its impact on the bias and variance trade-off.

4.1 Explicit Regularisation

Formally, explicit regularisation adapts the objective function to have the form

$$O(\mathbf{w}) = \ell(y, f(\mathbf{x}; \mathbf{w})) + \lambda R(\mathbf{w}), \tag{4.1}$$

where w are the model parameters. The term $R(\mathbf{w})$ is the *regulariser* or *regularisation term* and measures the complexity of the model, penalising highly complex models. The regularisation

[1]Deep learning theory literature often refers to this as *implicit bias* or *algorithmic bias*.

parameter λ measures the strength of the regularisation. How do we define the complexity of a model?

Consider a simple linear model where the variable Y is modelled by weighted linear combinations of the input features or

$$Y \approx \sum_i \mathbf{w}_i \, \tilde{\mathbf{x}}_i = \mathbf{w}^\intercal \, \tilde{\mathbf{x}}.$$

The model's complexity can be defined with respect to the entries of $\mathbf{w}$. If $\mathbf{w}_i$ is zero, the regressor represented by $\tilde{\mathbf{x}}_i$ is not involved in the model. Very naïvely, the sum of the model weights is inversely related to the model complexity; the more variables a model involves, the more flexibility the model has and a greater tendency to overfit. There are three main formulations of $R(\mathbf{w})$:

- *Least Absolute Shrinkage and Selection Operator* (LASSO) regularisation or L_1 regularisation is where $R(\mathbf{w}) = \sum_i \mathbf{w}_i$.
- Ridge regularisation or L_2 regularisation is where $R(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \mathbf{w}^\intercal \, \mathbf{w}$.
- *Elastic-net regularisation* [84], where both L_1 and L_2 regularisation terms are used.

4.2 Regularisation and Variance

Regularisation reduces the risk of overfitting by penalising overly complex or flexible models, but how does this relate to the bias-variance tradeoff? Reducing model flexibility will reduce the variance model, and as a side effect, the bias will increase. So regularisation trades variance for bias. We demonstrate the effect of regularisation by estimating the bias and variance of different models: the first is a linear classifier using cross-entropy loss, and the second is a ridge regression task. Finally, we explore why regularisation works by considering regularisation as a constrained optimisation problem.

4.2.1 Cross Entropy Demonstration

For this demonstration, logistic regression using an L_2-penalty term was trained on the IRIS data set [29]. The expected risk was decomposed, and the inverse regularisation strength parameter[2] C was varied. To take the expectation over $\mathcal{D}$, we performed an 80/20 train-test split and the training set was partitioned into five disjoint subsets.

Figure 4.1 shows the experiment results and that these results match what we expect; the expected risk has a U shape, a monotonically decreasing variance, and an increasing bias with increasing regularisation. As regularisation strength increases, model flexibility decreases,

[2]Due to scikit-learn's [57] implementation of logistic regression, as $C \rightarrow 0_+$ the stronger the regularisation. See: `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

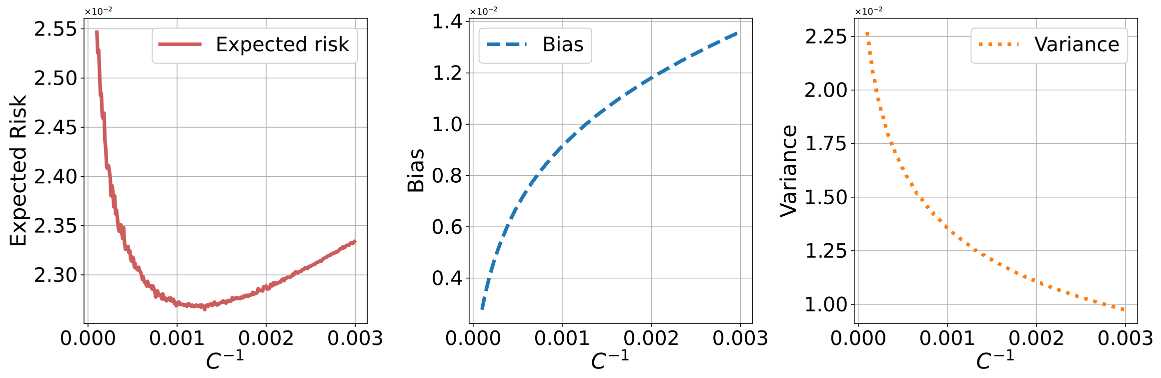

Figure 4.1: Bias-variance decomposition for L_2 regularised logistic regression on the IRIS data set. **(LEFT)** Expected risk. **(CENTRE)** Bias. **(RIGHT)** Variance.

causing variance to fall and bias to increase. The ideal regularisation strength $C^{-1} \approx 1.25 \times 10^{-3}$ ($C \approx 800$), which is a minimal amount of regularisation. We will now move on to the linear regression demonstration.

4.2.2 Ridge Regression Demonstration

Combining an L_2 regulariser with a linear model using the squared loss gives *ridge regression*. The bias and variance were estimated using a similar approach for a ridge regression model for the California housing data set [42]. Figure 4.2 shows the results of the experiments and that they broadly match our expected results, as we have a U-shaped risk curve and the variance monotonically decreases with increasing regularisation and the optimal amount of regularisation is $\lambda \approx 150$.

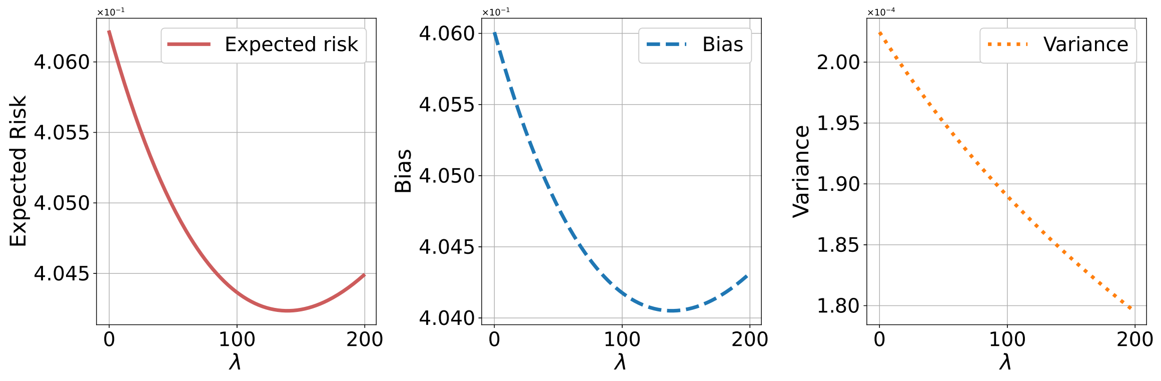

Figure 4.2: Bias-variance decomposition for ridge regression performed on the California housing data set. **(LEFT)** Expected Risk. **(CENTRE)** Bias. **(RIGHT)** Variance.

So what is going on? The variance is the easiest of the two to explain. Increasing the regularisation constrains the model and reduces its flexibility, so it will not be as sensitive to the underlying training data, and the variance decreases. However, the behaviour of the bias is a bit strange. The model becomes simpler with stronger regularisation, so the bias should increase as the amount of systematic error between the underlying generation process and the model increases. This is not the case in Figure 4.2. Instead of a monotonically increasing bias, we have a U shape. A linear model is a poor choice for this particular data set, so there is a high systematic error at all points. Regularisation decreases the model's expected risk due to the ridge regression's statistical properties. For any linear regression task, there always exists a regularisation constant $\lambda > 0$ with a lower squared loss than a non-regularised linear regression [28, 73].

4.3 Why Does Regularisation Work?

Equation (4.1) gives the general formula for explicit regularisation. We can connect it to the optimisation problem being solved. The regularisation term is a *Lagrange multiplier*[3], so Equation (4.1) can be expressed as the following optimisation problem:

$$\begin{array}{ll} \text{minimise} & \ell(y, f(\mathbf{x}; w)) \\ \text{subject to} & R(w) \leqslant c \end{array}$$

Returning to the function spaces considered and visualised in Chapter 2, the regularisation term restricts the model family $\mathcal{F}$ to a new family $\mathcal{R} \subset \mathcal{F}$, which satisfies the constrained optimisation problem. The model learnt will be simpler and have lower complexity, causing an increase in bias and a decrease in variance.

Figure 4.3 visualises the effect of regularisation. The function spaces $\mathcal{F}$ and $\mathcal{R}$ are added to the diagram to give a general notion of their size. As the regularisation strength increases, the function space $\mathcal{R}$ becomes smaller and further constrains the model, causing the model bias to continue to increase and the variance to continue to decrease.

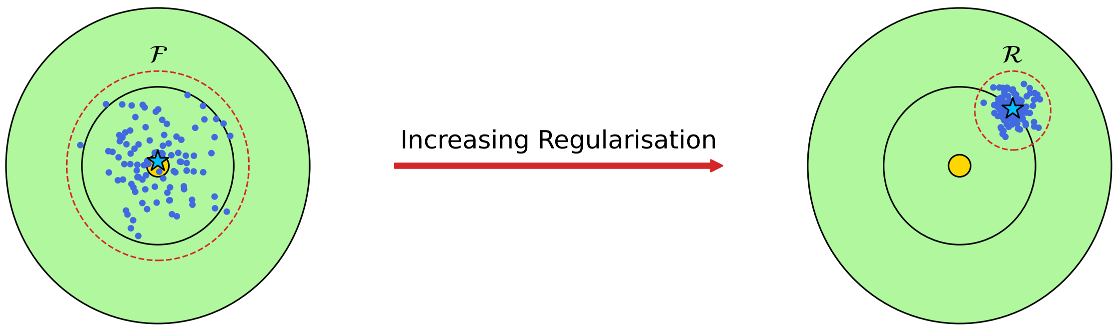

Figure 4.3: Visual demonstration of the effect of regularisation where the bullseye is the true value, the star is the cluster centroid, and each dot is a possible model output.

[3]This is a way to constrain the optimisation problem. See Boyd and Vandenberghe [12, Chapter 5] for more details.

4.4 Chapter Summary

In this chapter, we discussed regularisation through the lens of the bias-variance tradeoff. We provided empirical evidence as to the behaviour of the regularised models using both the squared and cross-entropy loss. We showed that as the amount of regularisation increases, the variance decreases. We concluded by providing a theoretical underpinning to the empirical results by exploring the constrained optimisation problem generated, which we connected back to SLT. In the next chapter, we discuss how we can use ensemble methods to combine models.

CHAPTER 5

Ensemble Methods

The ‘wisdom of the crowd’ is the idea that a diverse set of opinions are better than one. This idea is common throughout our society, from our parliamentary democracy to the jury system used in criminal trials. A famous example of this, especially among statisticians, is Francis Galton’s 1907 paper published in Nature [35], in which he recorded all the guesses of a cow’s weight people made in a county fair competition. He found that the median estimate had an error of only 1%, and the mean estimate was the ox’s weight. *Ensemble methods* aims to apply similar strategies to ML algorithms.

Instead of training a single model, we train a committee or *ensemble* of models. Each test point is fed into each ensemble member, and the outputs of all the models are combined. Ensemble members may predict real values for regression tasks, an output class, or a probability for classification tasks. These outputs must be merged somehow, with possible approaches being averaging, voting, or another probabilistic method. The underlying principle is every model will make some errors, but if we have multiple versions of the same model, each model makes different errors. By combining all these models’ predictions the errors will cancel out.

5.1 Why Do We Combine Models?

We shall first deal with regression tasks or combining real values. Jensen’s inequality states that for any convex function ϕ with a set of M reals $x_1, x_2, \dots x_M$ in the domain of ϕ we have

$$\phi\left(\sum_{i=1}^{M} a_i x_i\right) \leqslant \sum_{i=1}^{M} a_i \phi(x_i), \tag{5.1}$$

where $\sum_i a_i = 1$. We can contextualise this better if we connect this back to Galton’s experiment. Let x_i be the guesses of the cow’s weight and y the cow’s weight. If we choose

$\phi(x) = (x-y)^2$, and assume $a_i = \frac{1}{M}$ for all i by Equation (5.1) we have

$$(\bar{x}-y)^2 \leqslant \frac{1}{M}\sum_{i=1}^{M}(x_i-y)^2. \tag{5.2}$$

Another way to express this is that the squared error of the average estimate is less than the average squared error over all predictions. We can quantify how much better it is. By adding and subtracting a $\bar{x}$ inside the square on the right-hand side of Equation (5.2) and expanding the resulting expression, we get the following.

$$(\bar{x}-y)^2 = \frac{1}{M}\sum_{i=1}^{M}(x_i-y)^2 - \frac{1}{M}(x_i-\bar{x})^2. \tag{5.3}$$

In ensemble learning, this is known as the *ambiguity decomposition* and was first introduced by Krogh and Vedelsby [47].

For classification tasks *majority voting* is used, can we quantify the error we expect our model to get? If we have M independent voters, each with an error rate of ϵ, the number of errors K follows a *Binomial distribution* such that $K \sim B(M, \epsilon)$. This gives us

$$p_K(k) = \Pr(K=k) = \binom{M}{k}\epsilon^k(1-\epsilon)^{M-k}.$$

A vote error only occurs if a majority of the ensemble members are in error, so we have

$$\Pr(\text{vote error}) = \sum_{k \geqslant \lceil \frac{M+1}{2} \rceil} p_K(k) = 1 - F_K\Big(\Big\lceil \frac{M+1}{2} \Big\rceil - 1\Big).$$

Figure 5.1 plots this probability against the ensemble size, we can see how the probability of an ensemble error changes based on the value of M. If each model has only a 1% error rate, we can find that the chance of an ensemble error is almost surely zero if $M \geqslant 10$, so the more classifiers we include in our ensemble, the better. Importantly, this only occurs if each classifier is independent and has the same constant error rate.

We now understand theoretically why combining model outputs is beneficial, but how do we do this? There are two key approaches: the first is using a *parallel construction* where we split the data set into several training sets and train a model on each, and the second is to use *sequential construction* where the model is trained to correct the errors of another model. We shall look at *bagging*, a parallel construction method, and *boosting*, a sequential construction method.

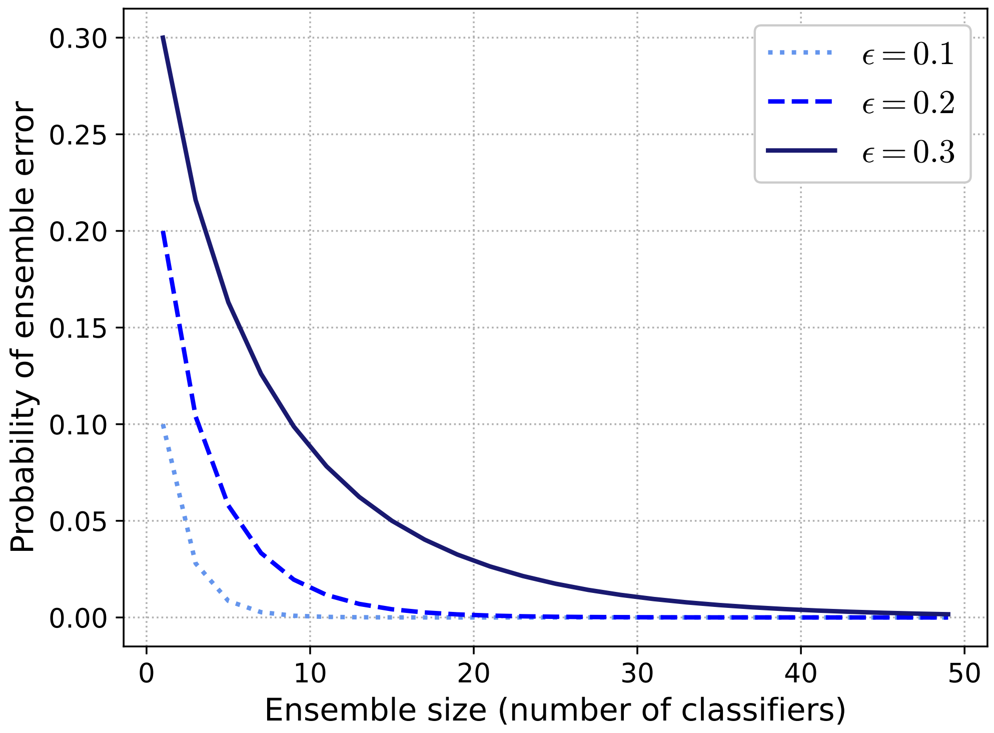

Figure 5.1: Effect on increasing the ensemble size on the probability of an ensemble error, we can see that as we increase the value of M, the probability of an ensemble error decreases. Importantly this trend only holds if the classifiers are independent.

5.2 Bagging

Breiman [14] introduced the meta-algorithm *bagging* or *bootstrap aggregating* for ensemble learning, which is the classic method for parallel ensembles. The idea is to take multiple *bootstrap samples* from the original training data set and use this to train a series of models. As bootstrapping gives different training sets, these differences are used to train a model and create a diverse ensemble of models. The models' outputs are combined using majority voting for classification problems or averaging for regression tasks. Algorithm 3 gives the pseudocode for a general bagging approach.

Algorithm 3 Bagging

```
Require: Training samples + labels $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, number of models $M$
  function BAGGING($\mathcal{D}$, $M$)
      $H \leftarrow \emptyset$
      for $i = 1, \dots, M$ do
          $\mathcal{D}' \leftarrow$ bootstrap sample generated from $\mathcal{D}$
          $h_i \leftarrow$ model trained using $\mathcal{D}'$
          $H \leftarrow H \cup \{h_i\}$
      return $H$
```

To demonstrate the effect of bagging as the ensemble size increases, bagging is performed on the

MNIST data set [49]. Figure 5.2 gives the classification error as a function of the ensemble size when bagging decision trees and Naïve Bayes classifiers. It is clear that increasing the number of decision trees in the ensemble causes the classification error to decrease, and it seems to level off at around 5%, which is much higher than under 1% Figure 5.1 suggests.

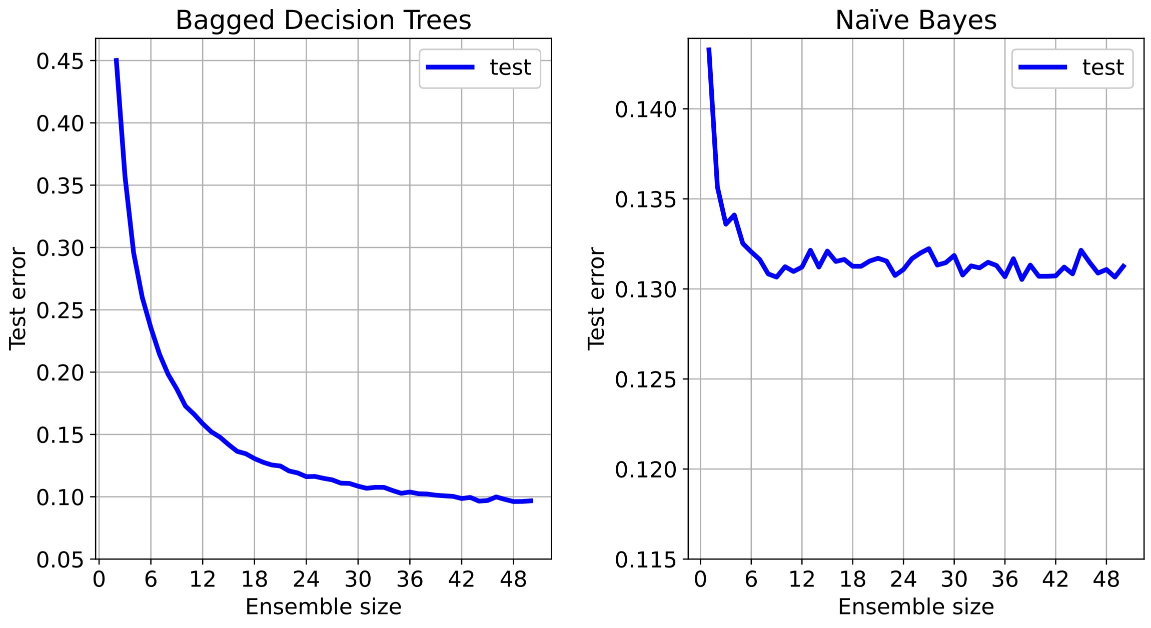

Figure 5.2: Example of bagging classifiers performed on the MNIST data set (50 bootstrap trials). **(LEFT)** Bagged decision tree classifiers. **(RIGHT)** Bagged na ive Bayes classifiers.

5.2.1 Analysing Bagging

Training data is being thrown away during the bagging process. Could this cause the mismatch between the theoretical value and the observed classification error? Let us consider the bootstrapping process on $\mathcal{D}_n$ in more detail. Since there are n training samples, each is selected with probability $\frac{1}{n}$ and not selected with probability $1 - \frac{1}{n}$. The bootstrap is of size n, so the chance an item is in the bootstrap sample is $1 - \left(1 - \frac{1}{n}\right)^n$. We can also consider what happens when the training data set becomes larger; the value tends to be around 0.632 and can be derived by taking the following limit.

$$\lim_{n\to\infty}\left\{1 - \left(1 - \frac{1}{n}\right)^n\right\} = 1 - e^{-1} \approx 0.632$$

This means that around 36.8% of the data will be excluded from a single bootstrap, but this doesn't tell the whole story.

During the bagging process, M bootstrap samples are taken. This means that each training sample has a 36.8% chance of being excluded from a single model and only a 0.368^M chance of being excluded from the entire ensemble. This probability decreases very quickly as the number of ensembles in the model increases, and when $M = 10$, there is less than a 1% chance an individual training sample is excluded.

As soon as the ensemble is reasonably large, all available training samples will almost certainly be considered. This is not the cause of the observed underperformance. What are the necessary conditions for the asymptotic behaviour in Figure 5.1? The key condition is model independence, so are the bagged decision classifiers independent? One way to measure the dependency is by plotting the *mutual information* [68] between the two classifiers, which is plotted in Figure 5.3.

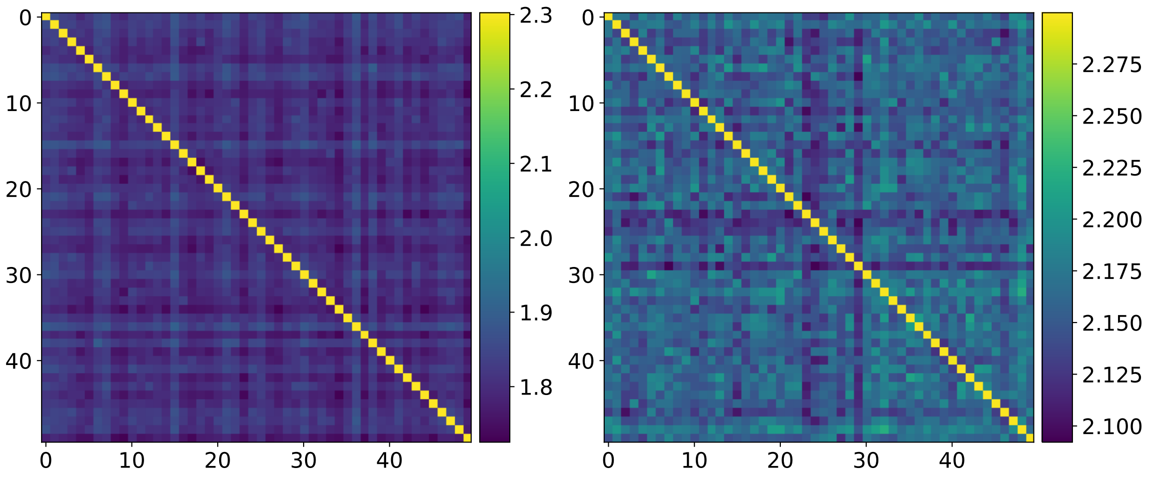

Figure 5.3: Dependency between bagged classifiers trained on MNIST. **(LEFT)** Bagged decision tree classifier. **(RIGHT)** Naïve Bayes classifier.

Looking at Figure 5.3, every single classifier depends on at least one other; this means the independent classifier assumption is completely invalid for the bagging ensembles presented and the theoretical results in Figure 5.1 is unattainable. Looking at the right sub-figure, there is even more correlation between models. This is because Naïve Bayes is a much more restricted model class than the class of decision tree classifiers or that it is a 'high bias' model. Thus, the models learnt are correlated together and often very similar. This is the real cause of the sub-optimal performance of both the bagged decision tree and bagged Naïve Bayes classifiers. Very similar models will make the same errors and will not cancel out. To improve this performance, we must find a way to increase the 'diversity' of the ensemble members. A possible way to do this for decision trees is to introduce a random or stochastic element to constructing the trees. The *random forest* algorithm is a way to do this.

5.2.2 Random Forest

Breiman [16] proposed the *random forest* algorithm, an ensemble method for decision trees, that improves classical bagging strategies by introducing an element of randomness into the construction of the decision tree. Randomness is introduced in two ways: (1) *bootstrap samples* (bagging) are used to generate the training examples for each model, and (2) a *random subset of the features* are used to perform the splitting. Algorithm 4 outlines the pseudocode for generating a random forest ensemble.

Algorithm 4 Random Forest

```
function RandomForest($\mathcal{D}$, features)
    $H \leftarrow \emptyset$
    for $i = 1, \ldots, M$ do
        $\mathcal{D}' \leftarrow$ bootstrap sample from $\mathcal{D}$
        $h_i \leftarrow$ GenerateRandomTree($\mathcal{D}', F$)
    return $H$
function GenerateRandomTree($\mathcal{D}'$, $F$)
    for all split point do
        Choose fraction $K$ of the remaining features
        Split on the best feature of those selected
    return generated tree
```

Generally, we set $K = \lceil \sqrt{d} \rceil$ where d is the number of input features, as this forces the tree to split more randomly. This randomness forces the ensemble members to be more diverse. It means that the errors of each member are likely to cancel out, leading to better classification performance than bagging decision tree classifiers.

Figure 5.4 gives the performance of a random forest of up to 50 trees on MNIST. Random forest outperforms simple bagging strategies on decision trees — the test error decreases to around 3%, which is lower than the 5% obtained by bagging decision trees and almost five times lower than bagged naïve bayes. The dependency heat map demonstrates why this occurs. The random forest classifiers are much less correlated, meaning the ensembled classifiers are much more different, and their errors are much more likely to cancel out. This observation raises an interesting point:

> *More diverse ensembles perform better.*

This idea will be explored more formally in Section 5.4.

5.3 Boosting

The second class of ensemble methods are sequential construction methods — where each classifier aims to correct the errors of the proceeding model. This approach is known as 'boosting'.

Boosting begins with a probability distribution defined over the training samples and indicates the relative weight each training sample should be given. At each step, a new classifier is trained, and if the training sample is predicted correctly or incorrectly is recorded. If the model is correct, the training sample's weight is decreased. For incorrect training samples, the

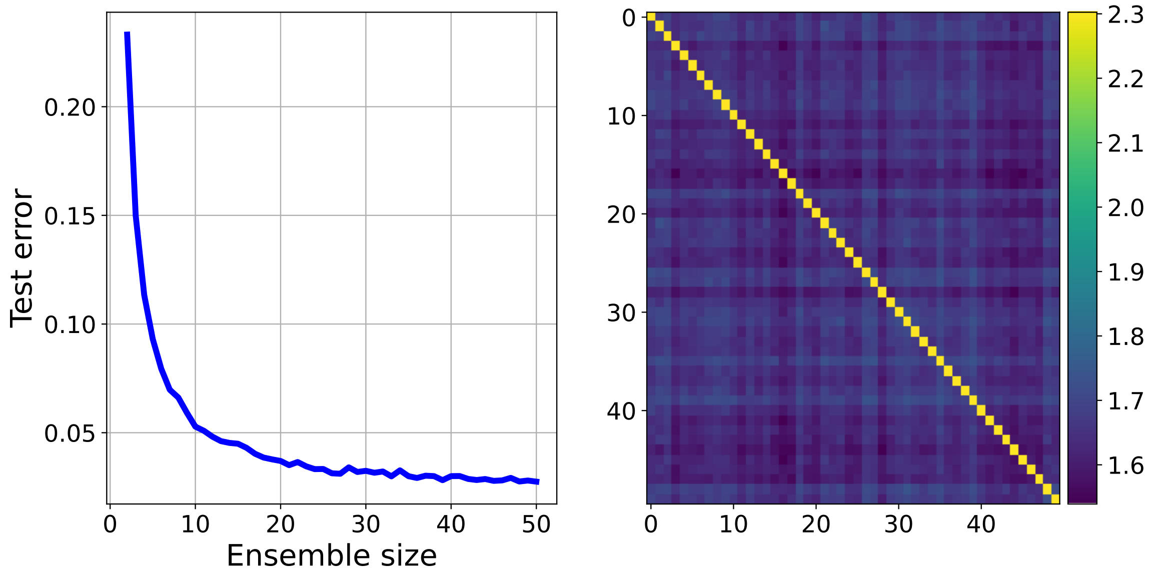

Figure 5.4: Random forest ensemble ($M \leqslant 50$) trained on the MNIST data set (100 bootstrap trials). **(LEFT)** Test error as the number of classifiers increased. **(RIGHT)** Dependency between the trained classifiers ($M = 50$).

weight is increased. This causes the next model to focus on those training samples that the proceeding model classified incorrectly.

Boosting is unique in the history of machine learning. It is one of the few models predicted to exist before the actual algorithm was created. In 1989, Kearns and Valiant [41] asked 'Can a set of weak learners create a single strong learner?' where a weak learner is a classifier that performs slightly better than random. Freund [31] and Schapire [67] demonstrated the answer was yes by developing initial versions of boosting as a constructive proof to answer the question Kearns and Valiant proposed. Freund and Schapire [32, 33] improved upon these initial approaches by introducing the *adaptive boosting* algorithm or *AdaBoost*. This is one of the most widely used boosting algorithms since its inception in 1995. Breiman [14] stated that 'AdaBoost with trees is best off-the-shelf classifier in the world'. Algorithm 5 gives the pseudocode for the AdaBoost algorithm.

Algorithm 5 AdaBoost

```
function AdaBoost
    Initialise probability distribution over the training set, $P_1(i) = \frac{1}{n}$ for all $i \leqslant n$
    for $t = 1, \dots M$ do
        Build classifier $h_t$ using $P_t(i)$
        $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
        for all $i \leqslant n$ do
            $P_{t+1}(i) \leftarrow P_t(i) \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$
        $P_{t+1} \leftarrow P_{t+1}/Z_t$
function AdaBoostPredict($\mathbf{x}$)
    return sign$\left(\sum_{t=1}^{M} \alpha_t h_t(\mathbf{x})\right)$
```

5.4 Why Do Ensemble Methods Work Well?

We have discussed the two main ensemble learning approaches and suggested that ensemble diversity is essential. Bagging creates a diverse ensemble by providing different training samples for each model. Boosting does this by forcing each model to become more accurate where the previous model was not. How do we quantify diversity and do we have a unified theory to explain why ensembles work and their connection to diversity? This was regarded as the holy grail of ensemble theory research, and only recently a possible solution was proposed by Wood et al. [80].

Krogh and Vedelsby [47] were the first to try it. They introduced the ambiguity decomposition for the squared loss, which we saw in Equation (5.3). More formally, we define it as follows.

Definition 13 (Ambiguity decomposition of the squared loss).
Given an ensemble of classifiers $\mathbf{q} = \{\mathbf{q}_i\}_{n=1}^{M}$ where $\bar{\mathbf{q}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{q}_i$, the *ambiguity decomposition* for the set of classifiers is

$$(\bar{\mathbf{q}}(\mathbf{x}_i) - y_i)^2 = \underbrace{\frac{1}{M}\sum_{i}^{n}(\mathbf{q}_i(\mathbf{x}_i) - y_i)^2}_{\text{average loss}} - \underbrace{\frac{1}{M}\sum_{i}^{n}(\mathbf{q}_i(\mathbf{x}_i) - \bar{\mathbf{q}}(\mathbf{x}_i))^2}_{\text{ambiguity}}.$$

This demonstrates that the ensemble members' ambiguity or diversity has the effect of reducing the ensemble loss. The trick to taking this further is to notice that Definition 13 is a special case of the bias-variance decomposition.

5.4.1 Bias-Variance-Diversity Decomposition

Wood et al. [80] propose the 'double decomposition' trick which leads to the *bias-variance-diversity* decomposition. The general structure of the decomposition is given in Figure 5.5 and involves performing two steps of the bias-variance decomposition.

$$\begin{tikzcd}[column sep=small]
 & & \mathbb{E}_{\mathcal{D}}\{\textbf{ensemble loss}\} \arrow[dl] \arrow[dr] & & \\
 & \mathbb{E}_{\mathcal{D}}\{\textbf{average loss} \arrow[dl] \arrow[dr] & - & \textbf{ambiguity}\} \arrow[dr] & \\
\overline{\textbf{bias}} & + & \overline{\textbf{variance}} & - & \textbf{diversity}
\end{tikzcd}$$

Figure 5.5: The double decomposition trick.

Let us begin by (again) considering the squared loss. We shall decompose the expected ensemble loss

$$\mathbb{E}_{\mathcal{D}}\left[(\bar{\mathbf{q}} - \mathbf{y})^2\right] \tag{5.4}$$

using this double descent trick. We first apply Definition 13 to Equation (5.4) to get

$$\mathbb{E}_{\mathcal{D}}\left[\frac{1}{M}\sum_{i}^{n}(\mathbf{q}_i-\mathbf{y})^2-\frac{1}{M}\sum_{i}^{n}(\mathbf{q}_i-\bar{\mathbf{q}})^2\right]. \tag{5.5}$$

Tidying up Equation (5.5) by pushing the expectations inwards gives

$$\frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{\mathcal{D}}\left[(\mathbf{q}_i-\mathbf{y})^2\right]-\mathbb{E}_{\mathcal{D}}\left[\frac{1}{M}\sum_{i=1}^{M}(\mathbf{q}_i-\bar{\mathbf{q}})^2\right]. \tag{5.6}$$

Applying Theorem 2.3 to Equation (5.6) yields

$$\frac{1}{M}\sum_{i=1}^{M}\left[(\mathbb{E}_{\mathcal{D}}[\mathbf{q}_i]-\mathbf{y})^2+\mathbb{E}_{\mathcal{D}}\left[(\mathbf{q}_i-\mathbb{E}_{\mathcal{D}}[\mathbf{q}_i])^2\right]\right]-\mathbb{E}_{\mathcal{D}}\left[\frac{1}{M}\sum_{i=1}^{M}(\mathbf{q}_i-\bar{\mathbf{q}})^2\right]. \tag{5.7}$$

Distributing the summations over Equation (5.7) gives the bias-variance-diversity decomposition of the squared loss:

$$\underbrace{\frac{1}{M}\sum_{i=1}^{M}(\mathbb{E}_{\mathcal{D}}[\mathbf{q}_i]-\mathbf{y})^2}_{\text{Average bias}}+\underbrace{\frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{\mathcal{D}}[(\mathbf{q}_i)-\mathbb{E}_{\mathcal{D}}[\mathbf{q}_i]]^2}_{\text{Average variance}}-\underbrace{\mathbb{E}_{\mathcal{D}}\left[\frac{1}{M}\sum_{i=1}^{M}(\mathbf{q}_i-\bar{\mathbf{q}})^2\right]}_{\text{Diversity}}.$$

So we can connect ensemble performance back to the bias-variance decomposition, which is helpful as it provides a framework to reason about model ensembles in a formalised way. This connection makes sense as a single model has the bias-variance decomposition, so it stands to reason that you can connect this to ensemble models. As there are multiple models, an additional term must give the effect of the different models. A similar decomposition can be done for Bregman divergence losses [see 80, Theorem 4]. The proof of this is irrelevant to the current discussion. For completeness, the generalised bias-variance-diversity decomposition is reproduced, as we will refer back to it later.

Theorem 5.1 (Wood et al. [80, Theorem 2]).
Consider a set of models $\{\mathbf{q}_1,\ldots,\mathbf{q}_M\}$, evaluated by a loss function ℓ. Assuming we have a bias-variance decomposition similar to Definition 1 of Wood et al. [80], the following *bias-variance-diversity decomposition* holds.

$$\mathbb{E}_{\mathcal{D}}[\ell(\mathbf{y},\bar{\mathbf{q}})]=\underbrace{\frac{1}{M}\sum_{i=1}^{M}\ell(\mathbf{y},\mathbf{q}_i^*)}_{\text{average bias}}+\underbrace{\frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{\mathcal{D}}[\mathbb{V}(\mathbf{q}_i^*,\mathbf{q}_i)]}_{\text{average variance}}-\underbrace{\mathbb{E}_{\mathcal{D}}\left[\frac{1}{M}\sum_{i=1}^{M}\mathbb{V}(\bar{\mathbf{q}},\mathbf{q}_i)\right]}_{\text{diversity}},$$

where $\mathbf{q}^*\overset{\text{def}}{=}\arg\min_{\mathbf{z}}\mathbb{E}_{\mathcal{D}}[\mathbb{V}(\mathbf{z},\mathbf{q})]$ and $\bar{\mathbf{q}}\overset{\text{def}}{=}\arg\min_{\mathbf{z}}\frac{1}{M}\sum_{i=1}^{M}\mathbb{V}(\mathbf{z},\mathbf{q}_i)$.

Theorem 5.1 assumes that the loss function can be decomposed into bias and variance terms. A notable exception is the *0-1 loss*. There is no bias-variance decomposition for the 0-1 loss [80, Theorem 9], so a bias-variance-diversity decomposition cannot exist. Hastie and James [37] argued there is no need to consider the bias and variance of the 0-1 loss. Instead, we only need to focus on its effect on the expected loss. This idea can be used to derive the *bias-variance-diversity effect decomposition* [80, Theorem 13], which describes the effect of each term on the expected loss of the ensemble.

5.4.2 The Effect of Ensemble Diversity

Bagging

Figure 5.6 shows the average bias, variance and diversity of the random forest and bagged decision tree ensembles as the number of classifiers increases to 50. Both ensembles exhibit a similar behaviour. As the number of ensembles increases, the expected ensemble loss decreases and, at the same time, the diversity of the model increases. Due to randomness in tree construction, the random forest has greater diversity as the ensemble size increases. This, in turn, is reflected in the lower expected risk of the random forest compared to the bagged decision trees.

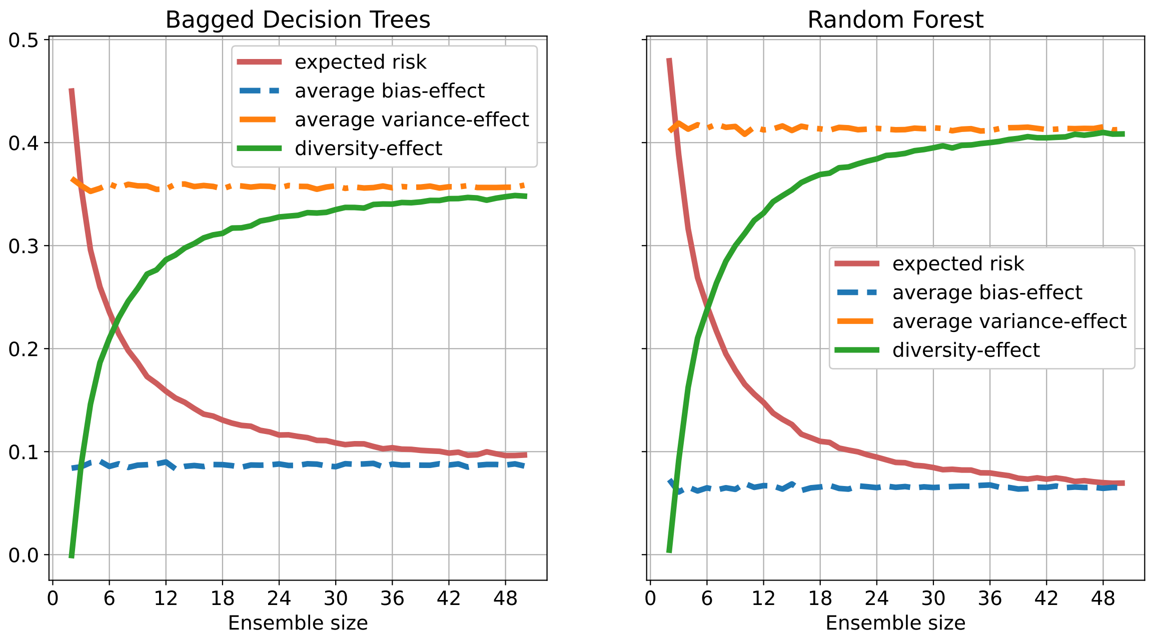

Figure 5.6: Bias-Variance-Decomposition of ensemble classifiers on MNIST (50 bootstrap trials). **(LEFT)** Bagged decision tree classifiers. **(RIGHT)** Random forest classifier.

If we look closely, it seems as if the diversity of the model levels off at the average variance. This results from the law of large numbers applied to the diversity term. As a consequence, the diversity of the model converges on the average variance as the ensemble size increases. So the effect of the diversity is to counteract or cancel out the variance of the ensembled model; this is a profound insight, as this allows us to conclude more general ideas about the effect of bagging.

Breiman [14] stated that 'a critical factor in whether bagging will improve accuracy is the stability of the procedure for constructing [the ensembled model]'. Breiman [15] explored this idea of stability further and attempted to classify known algorithms. In this context, a model is unstable if a small change in the training set can lead to a significant difference in the trained model. Another way of saying this is that bagging works best with high variance models such as deep decision trees or large neural networks as they are unstable and would work poorly with very stable models such as a k-NN classifier with 0-1 loss [26] and support vector machines [11].

This can be connected back to the bias-variance-diversity decomposition. During the bagging process, multiple bootstrap samples are taken from the training samples, which are then used to train a model. As the model family is unstable and each bootstrap sample is slightly different from the rest, the trained models will be different, and the ensemble diversity will be high. Highly unstable models with a high variance tend to have a low bias, and the diversity of models acts to cancel out this high variance which leaves behind the low bias. So bagging exploits the high variance of the model. This is why random forests outperform simply bagging decision tree classifiers, as random splitting causes random forests to be much more unstable.

Boosting

The abovementioned theory can be adapted to allow for weighted majority voting of the ensemble members [see 80, Appendix D.2]. This can then be used to estimate the bias-effect, variance-effect and diversity-effect for boosting algorithms. Figure 5.7 demonstrates the effect of boosting decision stumps[4] on the Mease data set [50] as the ensemble size varies.

There are some noticeable differences between the behaviour of parallel ensemble constructions (Bagging and Random Forest) compared to sequential approaches (AdaBoost and LogitBoost [see 34]). The bias and variance remain constant for bagging approaches but vary with ensemble size for boosting. This is due to the heterogeneity[5] of the ensembled classifiers, each classifier is trained to focus on a small subset of the training samples. Additionally, for boosting approaches, the diversity-effect can be greater than the variance-effect. This is a consequence of the boosting process. Each model is trained to correct the errors of the previous models, so differences between the models are encouraged as the models must correct the previous errors.

Whereas bagging the decrease in the error comes from the cancelling out of the variance term by increasing diversity The situation is more complicated for boosting. It relies on a tradeoff between bias, variance and diversity. However, it is possible to make a general statement about why boosting performs well. As the ensemble size increases, the bias-effect decreases as each

[4]A decision stump is a single layer decision tree or a decision tree with only an initial root node which immediately connects to two leaf nodes.

[5]An ensemble is *heterogeneous* if the ensemble members are from different model families and *homogeneous* if they are all from the same family.

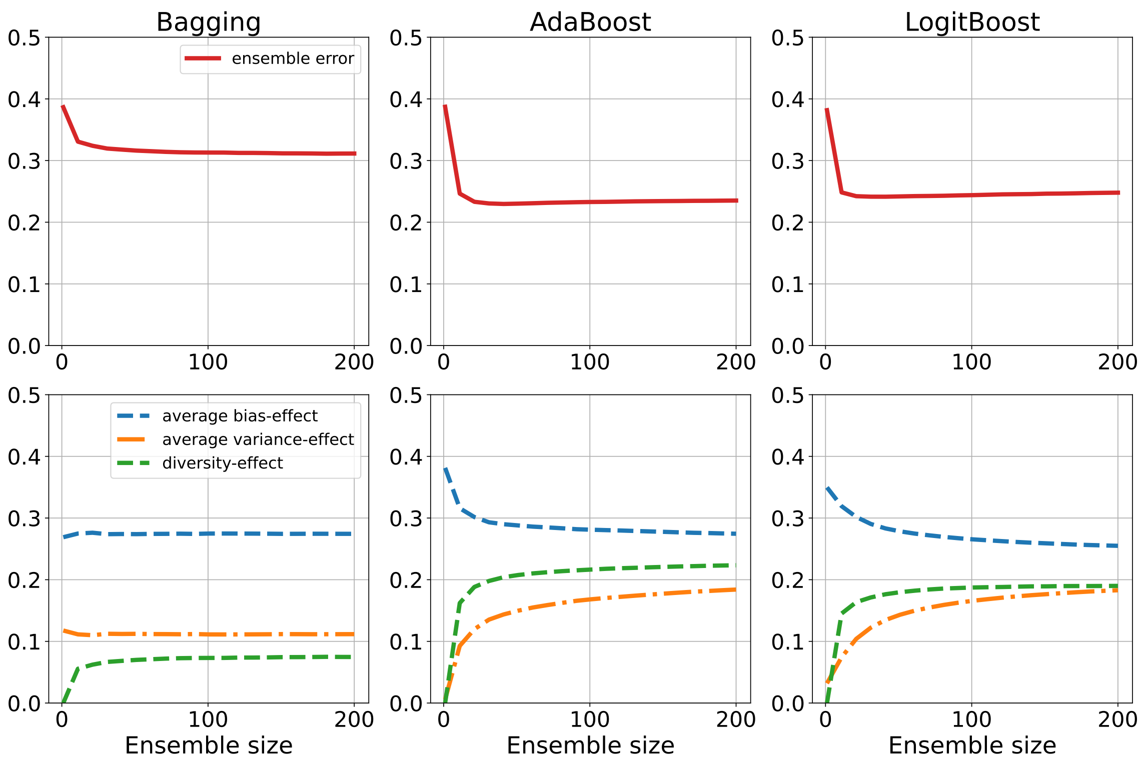

Figure 5.7: Bias-Variance-Diversity effect decomposition for boosting algorithms on Mease using decision stumps. Adapted from Wood et al. [80, Figure 18]

model aims to correct the previous, so the model complexity increases. Due to this increase in model complexity, the model's variance increases which is (approximately) cancelled out by the diversity-effect, leaving behind the model's bias, but all three terms are in tradeoff. So boosting works best when the ensembled models have a high bias (a weak classifier), and the effect of boosting is to decrease the overall bias of the model. This is consistent with the observations of Schapire [67] in his original proof.

5.5 Chapter Summary

We began this chapter by considering why we might want to combine the multiple models and gave a statistical and mathematical justification for the results. Next, we introduced bagging as a parallel ensemble construction technique and then improved upon it by introducing the random forest algorithm, which improves upon classical bagging. We then moved to sequential ensemble techniques, where AdaBoost was introduced and demonstrated its importance in the history of machine learning. Finally, we discussed ensemble theory and presented a unified theory to connect ensemble techniques to the bias-variance decomposition. In the next chapter, we will see where the classical theory breaks down by discussing recent literature that applies bias-variance theory to deep learning models.

CHAPTER 6

Bias, Variance and Deep Learning

So far, we have seen the classical understanding of a model's bias and variance. Bias monotonically decreases, and variance increases with increasing model complexity, yielding the classic U-shaped risk curve. Under this framework, the aim is to find a sweet spot of model complexity with the appropriate tradeoff between a model's bias and variance.

However, modern deep learning practice demonstrates that larger, deeper models with many hidden units are beneficial [45, 70, 83]. This contradicts the classical theory. These very large and complex models should have a very high variance, so they should overfit the data and not generalise well. This suggests that Geman et al. [36] were incorrect or need to be adapted. It becomes even stranger; the risk curve is not even U-shaped for some deep neural networks. It has the expected shape up to a point and then begins to drop off again. This is known as the 'double descent' phenomenon.

Geman et al. [36] were the first to study the bias and variance of neural networks and demonstrated that they follow the classical theory as width/depth increases. However, these experiments were limited as they only considered a simple network with four hidden layers. Neal et al. [56] were the first to empirically measure the bias and variance of modern neural network architectures; they demonstrated that the variance could decrease as the width increases toward a highly overparameterised regime. In this case, we say that a model is *overparameterised* if $\rho = \frac{p}{n} > 1$ where n is the number of training samples and p is the number of parameters the model has, or that the number of model parameters is much larger than the number of training samples. The *highly overparameterised* regime is where $\rho \gg 1$.

In a series of papers, Belkin et al. [8–10] studied the risk curves for various modern machine learning algorithms and proposed the *double descent* risk curve. They define it as the classical U-shape when $\rho < 1$, the risk begins to drop after this point. The point $\rho = 1$, where we enter the overparameterised regime, is called the *interpolation threshold*. Double descent has been shown to occur and described analytically in various regression and classification models.

6.1 Survey of Double Descent Literature

Advani and Saxe [1] explored the generalisation error's behaviour in high-dimensional neural networks. They demonstrated that a double descent curve could occur when training without stopping early. Additionally, they used random matrix theory[6] to describe the dynamics of both the generalisation error and the training error. This showed that the generalisation performance was worse on intermediate depth neural networks around $\rho = 1$ and that very large networks do not harm generalisation performance.

Nakkiran et al. [55] continues on the work of Belkin et al. [8] and shows that various deep learning models experience double descent, and it occurs as a function of both model size and the number of training epochs as compared to model size alone. Additionally, they demonstrate that sometimes artificial training noise must be added to observe a double descent curve and that increasing the number of training samples hurts generalisation performance for some regimes.

Several papers have explored the asymptotic behaviour of overparameterised models and have demonstrated useful theoretical results.

- Ba et al. [3] analytically described the asymptotic behaviour of a two-layer linear neural network[7] where the number of samples n, dimensions d and hidden units p all tend to infinity at a constant rate. They demonstrate that double descent [3, Fig. 1] occurs when the weights of the second model are optimised, and bias-variance decomposition of the network is also presented [3, Figs. 2 and 3].
- Concurrently Mei and Montanari [51] explored the generalisation error of random feature regressions and found asymptotic behaviour of both the training and generalisation errors using a similar random matrix theory approach to Ba et al. [3]. They similarly found that the model experiences a double descent curve, and the bias and variance were estimated.
- Hastie et al. [38] explored high-dimensional ridgeless linear regression, and the asymptotic behaviour of the generalisation error was found along with the model's bias and variance. Double descent was found to occur, and they suggested that overparameterisation is helpful for linear regression tasks. This paper is not directly related to deep learning, but literature has demonstrated an equivalence between neural networks and linear models [2, 21, 27, 40].

Each of these papers demonstrates that bias and variance are non-monotonic and experience a peak in specific regimes.

[6]This is a branch of maths concerning the properties of matrices whose elements are randomly drawn from a probability distribution [see 22].

[7]This is a neural network with no activation function between its hidden layer and final output logits.

But why exactly does double descent occur? Yang et al. [81] demonstrated that double descent in deep neural networks could be connected back to the bias-variance tradeoff. However, the classical theory of Geman et al. [36] must be adapted so that, as before, the bias is monotonically decreasing, but the variance is unimodal (bell-shaped), not monotonically increasing.

6.2 Unimodal Variance and Double Descent

Until Yang et al. [81], the relationship between double descent and deep neural networks was poorly understood, and many unanswered questions remained. First, double descent is hard to observe in a robust and predictable way [3, 55]. For modern deep learning architecture, sometimes artificial label noise must be added to reveal the double descent risk curve [55]. Second, there was no good explanation for why double descent occurs for deep neural networks. Yang et al. [81] solved both problems by proposing a *unimodal variance*, or that the bias remains monotonically decreasing with model complexity but that the variance rises to a peak and then steadily drops off again.

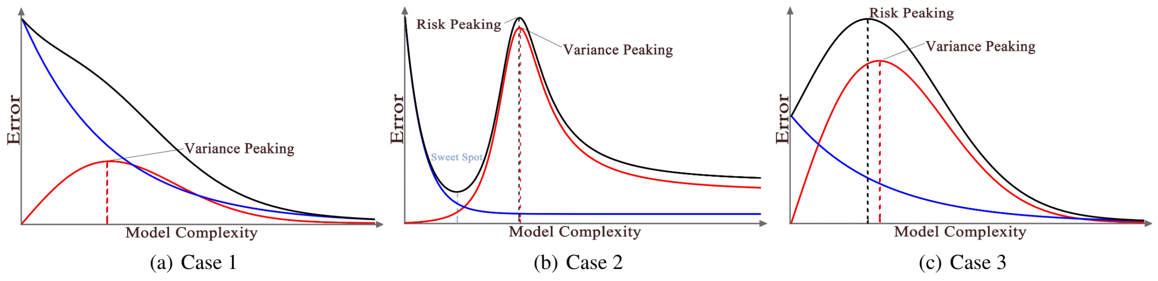

Figure 6.1: Typical cases of the expected risk curve (in black) for neural networks. Blue: squared bias. Red: variance. Figure borrowed from Yang et al. [81].

Figure 6.1 demonstrate the typical risk curves (shown in black) that are observed in neural networks. Of these, Figure 6.1a is the most common, and Figure 6.1b is the double descent curve. So in a sense, we can say that the bias-variance tradeoff still holds for modern deep learning architectures. Still, the underlying tradeoff and theory must be adapted in the manner Yang et al. [81] demonstrated.

This seems to contradict other theoretical work [3, 38, 51]. However, this difference is due to the approach to finding the generalisation error. There are two ways to describe the generalisation errors of the model: the random design and the fixed design. Each lead to a different bias-variance decomposition[8]. Ba et al. [3], Hastie et al. [38] and Mei and Montanari [51] use the fixed design, whereas Yang et al. [81] uses the random design, which is a more natural approach to take and leads to the same bias-variance decomposition included in this report.

[8]The particulars of these different approaches are irrelevant to the current discussion, but the details can be found in §2.1 of Yang et al. [81]

6.3 Empirical Results

Visualising the unimodal variance on a relatively simple network architecture is possible. We trained a fully connected deep neural network (DNN) with a single hidden layer and ReLU activation on the MNIST dataset for 200 epochs using SGD with momentum 0.9. The learning rate was 0.1, and a weight decay[9] of 5×10^{-4} was applied. Figure 6.2 shows the DNN's bias, variance and risk as a function of the network width.

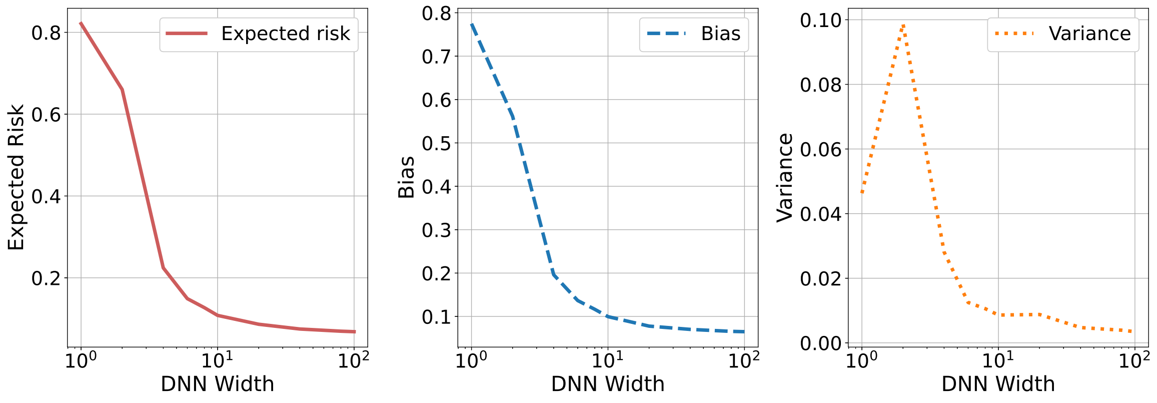

Figure 6.2: Expected risk, bias and variance of a fully connected network with one hidden layer and ReLU activation trained on squared loss. **(LEFT)** Expected risk. **(CENTRE)** Bias. **(RIGHT)** Variance.

So what can we see? As Geman et al. [36] suggested, the network's bias decreases monotonically as the width (number of hidden units) increases. This is because the more hidden units, the more complex and expressive representation can be encoded by the network or put another way, the network becomes more flexible, and the model complexity increases. As Yang et al. [81] predicted, the variance is unimodal, with a clear peak occurring when the network width is 2, decreasing monotonically from that point. The peak occurs here as this is around the interpolation threshold and is consistent with Belkin et al. [8–10].

The DNN's risk does not follow a double descent curve. Instead, it follows Case 1 of Figure 6.1. If we look at Figure 6.2, we can see why this occurs — the bias dominates the variance, so the bias is the main contributor to the risk and will be the main contributor to the risk curve's shape. This also gives us an insight into the nature of double descent. We can conclude there are two preconditions for double descent: (1) unimodal variance, and (2) the variance dominates the bias. These preconditions are evident in Case 2 of Figure 6.1 and Figures 2, 3(b) and 11 of Yang et al. [81].

[9]This means that an L_2 regularisation is applied to the network. Krogh and Hertz [46] demonstrated why this is beneficial.

6.4 Chapter Summary

In this chapter, we have discussed how bias-variance theory applies to state-of-the-art deep learning approaches. We began by demonstrating how the classical theory breaks down when considering modern deep learning architectures and then discussed the double descent risk curve that these architectures can experience. Then, we summarised the recent literature regarding double descent across various overparameterised regimes. Finally, we demonstrated that the behaviour of modern deep learning architectures could be explained by adapting classical theory so that the variance is unimodal. We have covered all the necessary material to answer the questions posed in Chapter 1. The final chapter presents our conclusions.

CHAPTER 7

Conclusion

Our goal for this report was to explore how bias-variance theory can be used to explain model behaviour. In this chapter, we first answer the four research questions posed in Chapter 1 and then discuss possible further work.

7.1 Summary of Results

7.1.1 Is the Bias-Variance Tradeoff a Helpful Way to Describe Model Behaviour?

The bias-variance tradeoff is a helpful way to describe model behaviour. It conveniently describes the vast majority of simple behaviour and gives a formal mathematical framework to allow us to reason about behaviour due to its connection to SLT. Additionally, it can be used to reason about more complicated cases: regularisation (Chapter 4), ensemble methods (Chapter 5), and overparameterised training regimes using large deep learning models (Chapter 6).

However, there are some fundamental limitations. Finding a bias-variance decomposition for several loss functions, most notably the 0–1 loss, is impossible, so a surrogate loss must be used instead. This can limit the types of models it is possible to reason with through this framework. Additionally, the bias-variance decomposition only gives the sources of generalisation error and the relative importance of each cause; it does not give a bound on the model's generalisation performance or say anything about the model's behaviour during training. We need other techniques to reason about these areas of model behaviour, and the bias-variance tradeoff is only a tiny part of a complete theory about model behaviour. Despite these limitations, the bias-variance tradeoff is among the most valuable ways to describe model generalisation. In most cases, the aim is to diagnose the cause of poor generalisation performance. The bias-variance tradeoff gives an intuitive way to do this.

7.1.2 How Can the Bias/Variance of a Model be Reduced?

It is possible to reduce the bias and variance of a model, Table 2.1 summarises possible approaches.

The variance of a model can be reduced in several ways:

- Reduce the number of training samples by either reducing the proportion of data left in the training split of the data set or early stopping when using an iterative-based optimiser.
- Applying explicit regularisation to the model reduces the model's variance by constraining the model class and forcing the trained model to be simpler. Recent literature on the theory of gradient-based optimisation methods suggests that using SGD-based optimisers act as a form of implicit regularisation and reduces the model's variance.
- Bagging a very unstable mode with a high variance causes the variance to be cancelled out by the ensemble diversity.

Reducing a model's bias is trickier but can be done in the following ways:

- Increase the number of training samples so the model has more data to update the weights for parameterised function approximators and to learn the mapping between the input and output spaces.
- Boosting can be performed, which has the general effect of reducing the bias of the final model. However, the impact of boosting the bias is more complicated.

7.1.3 When Does the Classical Bias-Variance Tradeoff Break Down?

Geman et al. [36] proposed the classical bias-variance tradeoff. This explanation for generalisation behaviour only applies to classical machine learning techniques using only a single model, which makes sense as this theory was originally proposed in 1992 and was very much at the early doors of the field. Chapters 2 and 4 demonstrated that it holds for a simple example of polynomial regression (Figure 2.8) and correctly explains the behaviour of explicit regularisation.

However, Geman et al. [36] cannot explain the behaviour of large or ensembled models, so the classical theory does break down in a sense. For ensemble methods, the classical theory holds for a single model, but the extension comes when diversity is introduced to describe the ensemble in addition to the individual member models. In a highly overparameterised regime, the classical theory must be adapted to a unimodal variance, but if $\rho < 1$, then the classical theory applies. Both cases suggest the classical bias-variance decomposition breakdown, but they are more accurately thought of as extensions of the classical theory and not the theory breaks down.

7.1.4 Does the Bias-Variance Tradeoff Hold for Modern ML Practice?

Chapter 6 explored recent literature on how the bias-variance tradeoff behaves for large deep learning models and overparameterised regimes. Recent theoretical and empirical work has attempted to explain the generalisation behaviour of deep networks, but as Yang et al. [81] demonstrated, the classical understanding of the bias-variance tradeoff must be adapted. So it appears that the bias-variance tradeoff is a possible way to explain why deep learning performs so well, or at least part of a potential tool to explain deep learning behaviour. However, it is very difficult to say anything more as this is an active area of research, and current literature only discusses relatively simple networks with few hidden layers.

7.2 Achievements

Before beginning this project, I had a general introduction to the concepts of both overfitting and underfitting, as well as the existence of bias and variance, through *COMP13212 Data Science* and *COMP24112 Machine Learning*. However, the mathematical background and the theoretical underpinnings of machine learning were novel to me as my previous course units tended to have a more practical focus. This was in addition to the ideas and literature of ensemble theory (Chapter 5) and deep learning theory (Chapter 6). With this in mind, I achieved the following during this project:

- Increased my understanding of machine learning and approaches to formalise notions of model behaviour using Statistical Learning Theory and the Empirical Risk Minimisation principle.
- Derived the bias-variance decomposition for both the squared, cross-entropy losses and the generalised bias-variance decomposition for Bregman divergences.
- I implemented code to calculate the bias-variance decomposition for the squared loss and KL divergence (see Chapter B for the code listings). I used the tool Wood et al. [80] developed for the bias-variance-diversity decompositions presented in Chapter 5.
- Explored recent literature to explore how the classical machine learning theory can be used to explain various machine learning approaches and techniques and gave empirical evidence to support this.

7.3 Reflection

All of the research questions posed in Chapter 1 have been answered, and there is sufficient theoretical underpinning and empirical evidence for the conclusions presented above. So I would conclude that this project has been a success. I have enjoyed researching and writing about bias-variance decompositions and the generalisation dynamics of machine learning models. During this project, I have become more confident reading research papers and developed my skills in writing proofs and mathematics in general. Completing this project has reinforced my desire to pursue postgraduate studies and eventually a PhD in machine learning theory — it has opened the door to the world of machine learning theory and the maths behind it.

My biggest challenge in the project was completing the KL divergence proof presented in Chapter 3. Initially, I attempted to use the decomposition presented by Heskes [39] as this was the original version of the proof. Unfortunately, the approach used in the paper was very unintuitive, and the article was poorly written, with the proof being challenging to follow and comprehend. The Bregman divergence proof presented is much simpler and easier to follow. I was aware of this approach towards the beginning of my project, and it would have been a more productive use of my time to focus on this approach from the start instead of Heskes' proof.

If I were to start this project again, I would have started the chapter and research on deep learning at an earlier opportunity. This would have allowed me to expand upon the empirical results given in the chapter and consider the case of more complex models to have an example of double descent risk curve with a unimodal variance and monotonically decreasing bias.

7.4 Future Work

7.4.1 Margin Losses

This project has only focused on distance-based loss functions such as squared and cross-entropy losses. Another type of loss function is *margin losses* such as the logistic loss and exponential loss. Wood et al. [79] analysed the bias-variance decomposition of margin losses. This can be used to study other machine learning algorithms, tasks and loss functions.

7.4.2 Bias, Variance and Deep Learning

Why deep learning approaches work and generalise well are open questions. Due to the dominance of deep learning, they are critical questions in ML theory. Current literature has only focused on very simple linear networks with few layers. Further work to analyse the asymptotic behaviour of bias/variance of non-linear networks as a function of increasing depth could help explain the generalisation performance of deep network architectures and lead to a richer theory underpinning empirical work.

7.4.3 A Unified Theory of Deep Learning

The bias-variance tradeoff describes the generalisation performance of machine learning algorithms. However, there are other questions we might want to know the answer to, such as: (1) can we calculate an upper bound on the generalisation error of the model, and (2) how quickly, if at all, will a model converge to a local minimum on the training samples. To answer these questions, different approaches and frameworks need to be used. Generalisation bounds can be estimated using statistical learning theory, including Vapnik-Chervonenkis (VC) dimensions [76, 77], Rademacher Complexities [7, 43, 44] and uniform stability [11, 53, 60]. Recent theoretical work exploring the training dynamics of gradient-based optimisers and their convergence properties [5, 18–20, 24] have increased our understanding of the training process. Each strand of current research explains an individual piece of machine learning, and combining these strands could lead to a unified theory to explain model behaviour and deep learning.

References

[1] Madhu S. Advani and Andrew M. Saxe. *High-dimensional dynamics of generalization error in neural networks*. 2017. arXiv: 1710.03667 [stat.ML]. preprint.

[2] Zeyuan Allen-Zhu, Yuanzhi Li and Zhao Song. *A Convergence Theory for Deep Learning via Over-Parameterization*. 2019. arXiv: 1811.03962 [cs.LG]. preprint.

[3] Jimmy Ba, Murat Erdogdu, Taiji Suzuki, Denny Wu and Tianzong Zhang. 'Generalization of Two-layer Neural Networks: An Asymptotic Viewpoint'. In: International Conference on Learning Representations. 2020.

[4] A. Banerjee, Xin Guo and Hui Wang. 'On the optimality of conditional expectation as a Bregman predictor'. In: *IEEE Transactions on Information Theory* 51.7 (2005), pp. 2664–2669. ISSN: 1557-9654.

[5] Raphaël Barboni, Gabriel Peyré and François-Xavier Vialard. 'On global convergence of ResNets: From finite to infinite width using linear parameterization'. In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh, Alekh Agarwal, Danielle Belgrave and Kyunghyun Cho. 2022.

[6] David G. T. Barrett and Benoit Dherin. *Implicit Gradient Regularization*. 2022. arXiv: 2009.11162 [cs.LG]. preprint.

[7] Peter L. Bartlett and Shahar Mendelson. 'Rademacher and Gaussian Complexities: Risk Bounds and Structural Results'. In: *Journal of Machine Learning Research* 3 (Nov 2002), pp. 463–482. ISSN: ISSN 1533-7928.

[8] Mikhail Belkin, Daniel Hsu, Siyuan Ma and Soumik Mandal. 'Reconciling modern machine learning practice and the bias-variance trade-off'. In: *Proc. Natl. Acad. Sci. U.S.A.* 116.32 (2019), pp. 15849–15854. ISSN: 0027-8424, 1091-6490.

[9] Mikhail Belkin, Siyuan Ma and Soumik Mandal. 'To Understand Deep Learning We Need to Understand Kernel Learning'. In: *Proceedings of the 35th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, 2018, pp. 541–549.

[10] Mikhail Belkin, Alexander Rakhlin and Alexandre B. Tsybakov. ‘Does data interpolation contradict statistical optimality?’ In: *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*. The 22nd International Conference on Artificial Intelligence and Statistics. PMLR, 2019, pp. 1611–1619.

[11] Olivier Bousquet and André Elisseeff. ‘Stability and Generalization’. In: *Journal of Machine Learning Research* 2 (Mar 2002), pp. 499–526. ISSN: ISSN 1533-7928.

[12] Stephen P. Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge: University Press, 2004. xiii+716. ISBN: 978-0-521-83378-3.

[13] L. M. Bregman. ‘The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming’. In: *USSR Computational Mathematics and Mathematical Physics* 7.3 (1967), pp. 200–217. ISSN: 0041-5553.

[14] Leo Breiman. ‘Bagging predictors’. In: *Mach Learn* 24.2 (1996), pp. 123–140. ISSN: 1573-0565.

[15] Leo Breiman. ‘Heuristics of Instability and Stabilization in Model Selection’. In: *The Annals of Statistics* 24.6 (1996), pp. 2350–2383. ISSN: 0090-5364. JSTOR: 2242688.

[16] Leo Breiman. ‘Random Forests’. In: *Machine Learning* 45.1 (2001), pp. 5–32. ISSN: 1573-0565.

[17] Tom Brown et al. ‘Language Models are Few-Shot Learners’. In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[18] Congliang Chen, Li Shen, Fangyu Zou and Wei Liu. ‘Towards practical Adam: non-convexity, convergence theory, and mini-batch acceleration’. In: *J. Mach. Learn. Res.* 23.1 (2023), 229:10411–229:10457. ISSN: 1532-4435.

[19] Xiangyi Chen, Sijia Liu, Ruoyu Sun and Mingyi Hong. *On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization*. 2019. arXiv: 1808.02941 [cs.LG]. preprint.

[20] Lénaïc Chizat and Francis Bach. ‘On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport’. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[21] Lénaïc Chizat, Edouard Oyallon and Francis Bach. ‘On Lazy Training in Differentiable Programming’. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox and R. Garnett. Vol. 32. Curran Associates, Inc., 2019.

[22] Romain Couillet and Zhenyu Liao. *Random Matrix Methods for Machine Learning*. Cambridge: Cambridge University Press, 2022. ISBN: 978-1-00-912323-5.

[23] Alex Damian, Eshaan Nichani and Jason D. Lee. *Self-Stabilization: The Implicit Bias of Gradient Descent at the Edge of Stability*. 2023. arXiv: 2209.15594 [cs.LG]. preprint.

[24] Soham De, Anirbit Mukherjee and Enayat Ullah. *Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration*. 2018. arXiv: 1807.06766 [cs.LG]. preprint.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Version 2. 2019. arXiv: 1810.04805 [cs.CL]. preprint.

[26] L. Devroye and T. Wagner. 'Distribution-free performance bounds for potential function rules'. In: *IEEE Transactions on Information Theory* 25.5 (1979), pp. 601–604. ISSN: 1557-9654.

[27] Simon S. Du, Xiyu Zhai, Barnabas Poczos and Aarti Singh. *Gradient Descent Provably Optimizes Over-parameterized Neural Networks*. 2019. arXiv: 1810.02054 [cs.LG]. preprint.

[28] R. W. Farebrother. 'Further Results on the Mean Square Error of Ridge Regression'. In: *Journal of the Royal Statistical Society. Series B, Methodological* 38.3 (1976), pp. 248–250. ISSN: 0035-9246.

[29] R. A. Fisher. 'The Use of Multiple Measurements in Taxonomic Problems'. In: *Annals of Eugenics* 7.2 (1936), pp. 179–188. ISSN: 2050-1439.

[30] Spencer Frei, Gal Vardi, Peter L. Bartlett and Nathan Srebro. *The Double-Edged Sword of Implicit Bias: Generalization vs. Robustness in ReLU Networks*. 2023. arXiv: 2303.01456 [cs.LG]. preprint.

[31] Yoav Freund. 'Boosting a weak learning algorithm by majority'. In: *Proceedings of the third annual workshop on Computational learning theory*. COLT '90. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 202–216. ISBN: 978-1-55860-146-8.

[32] Yoav Freund and Robert E. Schapire. 'A desicion-theoretic generalization of on-line learning and an application to boosting'. In: *Computational Learning Theory*. Ed. by Paul Vitányi. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 1995, pp. 23–37. ISBN: 978-3-540-49195-8.

[33] Yoav Freund and Robert E. Schapire. 'Experiments with a new boosting algorithm'. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML'96. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996, pp. 148–156. ISBN: 978-1-55860-419-3.

[34] Jerome Friedman, Trevor Hastie and Robert Tibshirani. 'Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors)'. In: *The Annals of Statistics* 28.2 (2000), pp. 337–407. ISSN: 0090-5364, 2168-8966.

[35] Francis Galton. 'Vox Populi'. In: *Nature* 75.1949 (1949 1907), pp. 450–451. ISSN: 1476-4687.

[36] Stuart Geman, Elie Bienenstock and René Doursat. ‘Neural Networks and the Bias/Variance Dilemma’. In: *Neural Computation* 4.1 (1992), pp. 1–58. ISSN: 0899-7667.

[37] Trevor Hastie and Gareth James. *Generalizations of the bias/variance decomposition for prediction error*. Stanford, CA: Dept. Statistics, Stanford Univ., 1997.

[38] Trevor Hastie, Andrea Montanari, Saharon Rosset and Ryan J. Tibshirani. *Surprises in high-dimensional ridgeless least squares interpolation*. 2020. arXiv: 1903.08560 [math.ST]. preprint.

[39] Tom Heskes. ‘Bias/Variance Decompositions for Likelihood-Based Estimators’. In: *Neural Computation* 10.6 (1998), pp. 1425–1433. ISSN: 0899-7667.

[40] Arthur Jacot, Franck Gabriel and Clement Hongler. ‘Neural Tangent Kernel: Convergence and Generalization in Neural Networks’. In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[41] Michael Kearns and Leslie Valiant. ‘Cryptographic limitations on learning Boolean formulae and finite automata’. In: *J. ACM* 41.1 (1994), pp. 67–95. ISSN: 0004-5411.

[42] R. Kelley Pace and Ronald Barry. ‘Sparse spatial autoregressions’. In: *Statistics & probability letters*. Statistics & Probability Letters 33.3 (1997), pp. 291–297. ISSN: 0167-7152.

[43] V. Koltchinskii and D. Panchenko. ‘Empirical Margin Distributions and Bounding the Generalization Error of Combined Classifiers’. In: *The Annals of Statistics* 30.1 (2002), pp. 1–50. ISSN: 0090-5364. JSTOR: 2700001.

[44] Vladimir Koltchinskii and Dmitriy Panchenko. ‘Rademacher Processes and Bounding the Risk of Function Learning’. In: *High Dimensional Probability II*. Ed. by Evarist Giné, David M. Mason and Jon A. Wellner. Progress in Probability. Boston, MA: Birkhäuser, 2000, pp. 443–457. ISBN: 978-1-4612-1358-1.

[45] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. ‘ImageNet Classification with Deep Convolutional Neural Networks’. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. Burges, L. Bottou and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[46] Anders Krogh and John Hertz. ‘A Simple Weight Decay Can Improve Generalization’. In: *Advances in Neural Information Processing Systems*. Vol. 4. Morgan-Kaufmann, 1991.

[47] Anders Krogh and Jesper Vedelsby. ‘Neural Network Ensembles, Cross Validation, and Active Learning’. In: *Advances in Neural Information Processing Systems*. Vol. 7. MIT Press, 1994.

[48] S. Kullback and R. A. Leibler. ‘On Information and Sufficiency’. In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. ISSN: 0003-4851. JSTOR: 2236703.

[49] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. ISSN: 1558-2256.

[50] David Mease and Abraham Wyner. 'Evidence Contrary to the Statistical View of Boosting'. In: *Journal of Machine Learning Research* 9.6 (2008), pp. 131–156.

[51] Song Mei and Andrea Montanari. *The generalization error of random features regression: Precise asymptotics and double descent curve*. 2020. arXiv: 1908.05355 [math.ST]. preprint.

[52] Microsoft. *Turing-NLG: A 17-billion-parameter language model by Microsoft*. Microsoft Research. 2020. URL: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/.

[53] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio and Ryan Rifkin. 'Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization'. In: *Adv Comput Math* 25.1 (2006), pp. 161–193. ISSN: 1572-9044.

[54] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[55] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak and Ilya Sutskever. 'Deep Double Descent: Where Bigger Models and More Data Hurt'. In: International Conference on Learning Representations. 2019.

[56] Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien and Ioannis Mitliagkas. *A Modern Take on the Bias-Variance Tradeoff in Neural Networks*. 2019. arXiv: 1810.08591 [cs.LG]. preprint.

[57] F. Pedregosa et al. 'Scikit-learn: Machine Learning in Python'. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[58] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee and Luke Zettlemoyer. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 [cs.CL]. preprint.

[59] David Pfau. *A Generalized Bias-Variance Decomposition for Bregman Divergences*. 2013.

[60] Tomaso Poggio, Ryan Rifkin, Sayan Mukherjee and Partha Niyogi. 'General conditions for predictivity in learning theory'. In: *Nature* 428.6981 (6981 2004), pp. 419–422. ISSN: 1476-4687.

[61] Lutz Prechelt. 'Early Stopping — But When?' In: *Neural Networks: Tricks of the Trade: Second Edition*. Ed. by Grégoire Montavon, Geneviève B. Orr and Klaus-Robert Müller. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, pp. 53–67. ISBN: 978-3-642-35289-8.

[62] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei and Ilya Sutskever. *Language Models are Unsupervised Multitask Learners*. OpenAI, 2019.

[63] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li and Peter J. Liu. *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. 2020. arXiv: 1910.10683 [cs.LG]. preprint.

[64] Noam Razin, Asaf Maman and Nadav Cohen. ‘Implicit Regularization in Hierarchical Tensor Factorization and Deep Convolutional Neural Networks’. In: *Proceedings of the 39th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, 2022, pp. 18422–18462.

[65] Mark Ried. *Meet the Bregman Divergences*. Inductio ex Machina. 2013. URL: https://mark.reid.name/blog/meet-the-bregman-divergences.html.

[66] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. In collab. with Ming-Wei Chang. Fourth edition., Global edition. Pearson series in artificial intelligence. Harlow: Pearson, 2022. ISBN: 978-1-292-40113-3.

[67] Robert E. Schapire. ‘The strength of weak learnability’. In: *Mach Learn* 5.2 (1990), pp. 197–227. ISSN: 1573-0565.

[68] C. E. Shannon. ‘A mathematical theory of communication’. In: *The Bell System Technical Journal* 27.3 (1948), pp. 379–423. ISSN: 0005-8580.

[69] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper and Bryan Catanzaro. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. 2020. arXiv: 1909.08053 [cs.CL]. preprint.

[70] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. preprint.

[71] Samuel L. Smith, Benoit Dherin, David Barrett and Soham De. ‘On the Origin of Implicit Regularization in Stochastic Gradient Descent’. In: International Conference on Learning Representations. 2022.

[72] Shaden Smith et al. *Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model*. 2022. arXiv: 2201.11990 [cs.CL]. preprint.

[73] C. M. Theobald. ‘Generalizations of Mean Square Error Applied to Ridge Regression’. In: *Journal of the Royal Statistical Society. Series B, Methodological* 36.1 (1974), pp. 103–106. ISSN: 0035-9246.

[74] Romal Thoppilan et al. *LaMDA: Language Models for Dialog Applications*. 2022. arXiv: 2201.08239 [cs.CL]. preprint.

[75] V. Vapnik. ‘Principles of risk minimization for learning theory’. In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS’91. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, pp. 831–838. ISBN: 978-1-55860-222-9.

[76] V. N. Vapnik and A. Ya. Chervonenkis. ‘On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities’. In: *Theory Probab. Appl.* 16.2 (1971), pp. 264–280. ISSN: 0040-585X.

[77] Vladimir Naumovich Vapnik. *The nature of statistical learning theory*. New York: Springer, 1995. ISBN: 978-1-4757-2440-0.

[78] Eric W. Weisstein. *Invertible Matrix Theorem*. 2020. URL: https://mathworld.wolfram.com/InvertibleMatrixTheorem.html.

[79] Danny Wood, Tingting Mu and Gavin Brown. ‘Bias-Variance Decompositions for Margin Losses’. In: *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*. Ed. by Gustau Camps-Valls, Francisco J. R. Ruiz and Isabel Valera. Vol. 151. Proceedings of Machine Learning Research. PMLR, 2022, pp. 1975–2001.

[80] Danny Wood, Tingting Mu, Andrew Webb, Henry Reeve, Mikel Lujan and Gavin Brown. *A Unified Theory of Diversity in Ensemble Learning*. 2023. arXiv: 2301.03962 [cs.LG]. preprint.

[81] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt and Yi Ma. ‘Rethinking Bias-Variance Trade-off for Generalization of Neural Networks’. In: *Proceedings of the 37th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, 2020, pp. 10767–10777.

[82] Yuan Yao, Lorenzo Rosasco and Andrea Caponnetto. ‘On Early Stopping in Gradient Descent Learning’. In: *Constr Approx* 26.2 (2007), pp. 289–315. ISSN: 1432-0940.

[83] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht and Oriol Vinyals. *Understanding deep learning requires rethinking generalization*. 2017. arXiv: 1611.03530 [cs.LG]. preprint.

[84] Hui Zou and Trevor Hastie. ‘Regularization and variable selection via the elastic net’. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. ISSN: 1467-9868.

APPENDIX A

Large Language Models (LLMs) Data

Year	Model name	Num. Parameters
Oct 2018	ELMo [58]	93.6M
Feb 2019	GPT-2 [62]	1.5B
May 2019	BERT [25]	340M
Sep 2019	Megatron-LM [69]	8.3B
Jan 2020	T5 [63]	11B
Feb 2020	Turing-NLG [52]	17.2B
June 2020	GPT-3 [17]	173B
May 2021	LaMBDA [74]	137B
Oct 2021	Megatron-Turing NLG [72]	530B

Table A.1: This is a tabular version of the data which was plotted in Figure 1.1

APPENDIX B

CODE LISTINGS FOR BIAS-VARIANCE DECOMPOSITIONS

To make the code modular, the `LossFunction` class was written as an abstract base class. To facilitate type hinting and autocompletion, `BiasVarianceResult` was typed as a named tuple. This allowed the experiment code to be loss function agnostic and gave better tooling support for autocompletion using type hints. For each loss function, we simply override the `calculate_bias_variance` method to implement the decomposition for each loss function.

Python Listing 1: Loss Function Abstract Class

```python
from typing import NamedTuple
from abc import ABC
from abc import abstractmethod
import numpy.typing as npt

class BiasVarianceResult(NamedTuple):
    expected_risk: npt.NDArray
    bias: npt.NDArray
    var: npt.NDArray


class LossFunction(ABC):
    @classmethod
    @abstractmethod
    def calculate_calculate_bias_variance(
        cls, true: npt.NDArray, pred: npt.NDArray
    ) -> BiasVarianceResult:
        ...
```

Python Listing 2: Squared Loss Decomposition

```python
import numpy as np
import numpy.typing as npt

class SquaredLoss(LossFunction):
    @classmethod
    def calculate_calculate_bias_variance(cls, y: npt.NDArray, f: npt.NDArray)
    ↪ -> BiasVarianceResult:
        """
        Calculates the bias-variance decomposition for the squared loss.
        :param y: 1 x N matrix of the test label
        :param f: D x N matrix of model predictions
        :return: named tuple of (expected_risk, bias, var)
        """
        Ef = np.mean(f, axis=0)
        expected_risk = ((f - y) ** 2).mean()
        avg_bias = np.mean((Ef - y) ** 2)
        Ef2 = np.mean(f**2, axis=0)
        avg_var = np.mean(Ef2 - Ef**2)

        return BiasVarianceResult(expected_risk=expected_risk, bias=avg_bias,
        ↪ var=avg_var)
```

Python Listing 3: KL Divergence Decomposition

```python
import numpy as np
import numpy.typing as npt
from scipy.special import rel_entr

class KLDivergence(LossFunction):
    @classmethod
    def calculate_expected(cls, outputs: npt.NDArray) -> npt.NDArray:
        """
        Calculates the expected model.
        :param outputs: Raw models output as D x N x K
        :return: expected model based on input as an K x N matrix
        """
        # Numerator - exp(E[ln(p_i(c)0)] this yields a N x K matrix
        numerator: npt.NDArray = np.exp(np.mean(np.log(outputs), axis=0))

        # Denominator - sum(exp(E[ln(p_i(c))]) where sum is over the K
        denominator: npt.NDArray = numerator.sum(axis=0)

        return numerator / denominator

    @classmethod
    def calculate_calculate_bias_variance(
        cls, pi: npt.NDArray, pi_0: npt.NDArray
    ) -> BiasVarianceResult:
        """
        Calculates the bias variance decomposition.
        :param pi_0: This is the one-hot-encoded true
            values (each column is a distribution) so
            is an K x N matrix
        :param pi: This is the output distribution
            from the model D x K x N matrix
        :return: BiasVarianceResult
        """
        pi_hat: npt.NDArray = cls.calculate_expected(pi)

        # Now get bias by averaging over the datapoints
        # E[K(p0 || hat{p})] for a individual datapoint i <= N
        bias = rel_entr(pi_0, pi_hat).sum(0).mean()

        # Variance - Ex[ET[K(hat{p} || p)]]
        variance = rel_entr(pi_hat, pi).sum(1).mean(0).mean()

        # Expected risk - Ex[ET[K(p0 || p)]]
        expected_risk = rel_entr(pi_0, pi).sum(1).mean(0).mean()

        return BiasVarianceResult(expected_risk=expected_risk, bias=bias,
         ↪  var=variance)
```