Active Learning with Gaussian Processes for Photometric Redshift Prediction

Jakub Leszek Nabaglo

A thesis submitted in partial fulfilment of the degree of
Bachelor of Philosophy (Honours) – Science

Australian National University

October 2017

Except where otherwise indicated, this thesis is my own original work.

Jakub Leszek Nabaglo
26 October 2017

This research made use of Astropy [3] and scikit-learn [19].

Abstract

It is expensive to obtain the spectrum needed to accurately calculate the redshift of an astronomical object. We often estimate an object's redshift using cheaper photometric measurements. Spectra are required to generate labels in this regression problem. Large training sets are hence costly, limiting their accuracy and range, whereas unlabelled samples are abundant.

We develop a regressor based on Gaussian processes, and apply active learning to select samples for labelling based on their potential for improving the model's accuracy. This permits us to improve redshift predictions at a reduced cost.

Acknowledgements

English is not powerful enough to express my gratitude to Cheng Soon Ong. His encouragement propelled me through this year, his questions inspired much contemplation, and the depth of his knowledge never ceased to amaze. I could not have wished for a better supervisor.

On many occasions I encroached on the time of Christian Wolf, whose enthusiasm for astronomy is truly contagious. I always left his office with relevant insight and domain knowledge, the kind of which we often overlook in data science.

I must thank my parents, Ilona and Leszek; my siblings, Agata and Andrew; and my girlfriend Lily. They supported me through many years of study, and through the peaks and valleys of life.

Finally, I am indebted to Matthew Alger, for his advice, careful proofreading, and assistance taming truculent software.

Contents

Nomenclature

$\mathbb{E}$	Expectation
$\mathbb{N}$	Natural numbers (non-negative integers)
$\mathbb{R}$	Set of real numbers
$\mathbb{R}_{\geq 0}$	Set of non-negative real numbers
$\mathbb{Z}_{>0}$	Positive integers
ϕ	Feature function
$\mathbf{t}$	Set of training example counts
$\mathbf{v}$	Kernelised weights vector
$\mathbf{w}$	Weights vector
$\mathbf{x}$	The input to predict for
$\mathbf{x}'_i$	The input for the i^{th} testing point
$\mathbf{x}_i$	The input for the i^{th} training point
$\mathbf{y}$	Training label vector
$\mathcal{D}$	Training set
$\mathcal{D}'$	Testing set
$\mathcal{E}$	Space of error functions
$\mathcal{F}$	Space of original functions
$\mathcal{GP}$	Gaussian process
$\mathcal{N}$	Normal distribution
$\mathcal{P}$	Powerset
$\mathcal{T}$	Space of training example count sets
$\mathcal{X}$	Predictor input space
$\mathcal{Y}$	Predictor output space
δ_i	See section 3.1
ℓ	Learning curve

$\mathfrak{D}$	Space of training and testing sets	
$\mathfrak{F}$	Space of models	
Γ	The gamma function	
$\hat{f}$	Predictor	
$\hat{f}_c$	Candidate predictor	
$\hat{y}'_i$	The prediction for the i^{th} testing point	
$\hat{\mathcal{F}}$	Space of original predictors	
λ	Regularisation constant	
$\langle \cdot, \cdot \rangle_{\text{F}}$	Frobenius inner product	
FVU	See section 3.1.1	
μ	Mean	
Φ	Feature matrix	
Π	Space of recommenders	
π	A recommender returning a set of recommendations	
Σ	Covariance matrix	
σ	Deviation	
σ_n^2	Noise variance	
τ	The ratio of a circle's circumference to its radius	
var	Variance	
ε_i	The energy of i^{th} state of a physical system	
ϖ	A recommender returning a score	
c	Speed of light in a vacuum	$299\,792\,458\,\text{m}\,\text{s}^{-1}$
D	A distance	
d	Dimensionality of our data	
d'	Dimensionality of our features	
dz bias	See section 3.1.3	
dz deviation	See section 3.1.4	
dz_i	See section 3.1	
E	Loss function	
e	An error function	
F	Model	

f	Original function	
g	The photometric g band	
H	Hilbert space	
h	Distribution of the label noise	
H_0	Hubble parameter	$71.9^{+2.4}_{-3.0}\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$
i	Indexing variable	
i	The photometric i band	
I_A	Intensity of an object A	
I_a	Intensity in the band a	
j	Indexing variable	
K	Kernel covariance matrix of X with itself	
k	Kernel	
K_*	Kernel covariance matrix of X with $\mathbf{x}$	
k_B	The Boltzmann constant	$1.380\,648\,52(79) \times 10^{-23}\,\mathrm{J\,K^{-1}}$
K_ν	The modified Bessel function of the second kind	
K_{**}	Kernel covariance matrix of $\mathbf{x}$ with itself	
m_A	Apparent magnitude of an object A	
m_a	Apparent magnitude in the band a	
n	Number of training samples	
n'	Number of testing samples	
P	Pool of unlabelled data	
p	Probability	
r	Size of the set of training example counts	
r	The photometric r band	
s_i	The i^{th} state of a physical system	
t	Step size	
u	The photometric u band	
v	A velocity	
X	Training input matrix	
y'_i	The label of the i^{th} testing point	
y_i	The label of the i^{th} training point	

z Redshift

z The photometric z band

z_{phot} Photometric redshift

z_{spec} Spectroscopic redshift

fraction of dz outliers See section 3.1.5

mean dz error See section 3.1.2

Chapter 1

Introduction

A redshift is the change in the wavelength of a photon emitted by an object that is moving away from the observer. When applied to the visible spectrum, a redshift makes an object appear more red. Redshifts are important in astronomy, as they are used for a variety of tasks. Most importanly, they permit us to study the evolution of the universe.

Currently, redshifts are found using spectroscopy. A spectroscope is applied to a galaxy, and its spectral profile is measured. From these measurements, we can isolate the emission and absorption lines characteristic of known elements, which are then compared against their baseline to obtain the redshift. This method is problematic as it requires a spectroscope to be pointed at each galaxy whose redshift we wish to measure. The number of galaxies in the universe is too great for this method to be a practical way of surveying them.

Instead, we can utilise photometric measurements, which counts the received light towards one of a small number of colours, much like a digital camera sees the visible spectrum through red, green, and blue. [chapter 2] These measurements can be used to predict the redshift. [section 3.2]

We wish to improve the accuracy of photometric redshift models with active learning. To do so, we develop a fast discriminative predictor through approximating Gaussian process regression. [section 3.9] It is able to return confidence intervals for its predictions.

These confidence intervals can then be used to find samples whose addition into the training set would be helpful. They can then be labelled selectively, saving resources. [chapter 4]

1.1 Redshift

Redshift is a property of all signals we observe from astronomical objects. It is often not easy to measure, but finding it permits us to learn important facts about the object being observed. It is the phenomenon due to which light waves emitted by a distant object may be 'stretched' out: their wavelengths are increased. Distant galaxies whose light is subject to redshift appear tinted red since that's the colour on the high-wavelength end of the visible spectrum.

There are three main causes of redshift. The main cause is the expansion of the universe. As the universe expands, space itself physically expands. Since light is a wave travelling through space, that wave also expands, increasing its wavelength. The light is thus redshifted.

The ratio of the size of the universe now to its size when the light we receive was emitted is $z + 1$, where z is the redshift of the object. Looking at high-redshift objects, we look into the past at a universe that was significantly smaller. Unsurprisingly, the universe appears much denser at higher redshifts, as a remnant of the earlier stages of the big bang. Similarly, large distances contain more quasars and more brighter objects. Redshifts are hence crucial in our study of the evolution of the universe and of the laws of physics.

Redshifts can also be caused by the Doppler effect. This is the phenomenon by which waves produced by an object that is moving away from the observer appear to have a longer wave-

length. Consider an ambulance driving away from you: the tone of the siren appears lower than if the ambulance was stationary. In a similar way, a galaxy moving away from us appears redder. There exists a converse effect, affecting objects moving towards the observer, analogously termed *blueshift*.

The redshift of an object is then directly correlated to its velocity perpendicular to the line of sight from the observer. For a small redshift z, the line-of-sight velocity is

$$v \approx cz,$$

where c is the speed of light constant.[1]

This permits us to approximate the distance from the observed object. Due to the expansion of space, distant objects are moving away from us at a velocity proportional to their distance. For a velocity v, that distance is

$$D \approx \frac{v}{H_0} \approx \frac{cz}{H_0},$$

where H_0 is the Hubble parameter, equalling approximately $71.9\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$. For faraway objects then, the redshift is hence proportional to distance.

The mass of an object is correlated to its luminosity[2]. The luminosity can be approximated from the brightness using the object's distance. Hence, computing the distance of an object from the redshift permits us to approximate its size.

However, for some objects, we can approximate the size more directly. For closer objects, we can measure their apparent radius as the angle they span on the sky. Using trigonometry, we can then compute the actual radius of such an object from its apparent radius and redshift-derived distance.

For closer galaxies, we are able to observe the movements of stars inside the galaxy. When a star orbits the centre of a galaxy, the internal motion can cancel or amplify the star's line-of-sight velocity. Measuring the stars' velocities from redshift hence permits us to to study galaxy dynamics.

Gravitational redshift is the final mechanism causing redshift, usually negligible in comparison to the others. Due to general relativistic effects, light escaping a gravitational field experiences redshift.

[1] For higher redshifts, the formula must account for special relativistic effects.

[2] The luminosity is its total amount of light emitted.

Chapter 2

Photometry

Photometry is a technique in astronomy, where we measure the amount of light that reaches a telescope from an object. The wavelength information is ignored, although filters may be used to discard some light, giving us limited wavelength information. [23]

Photometry is analogous to digital photography. Our cameras use three filters (red, green, and blue) to separate colour information. They then measure he amount of light reaching the sensor. After processing, they yield three colour values per pixel.

Spectroscopy is a related technique that not only measures the amount of received, but also measures its wavelength. This yields a full spectrum of the object, instead of a few colours. It is used to accurately compute the redshift of an object: the spectra of distant objects are compared to those of nearby objects, and the offset between them is the redshift.

However, spectroscopy is expensive since a telescope can only measure the spectrum of a few objects at a time. This would make it extremely time-consuming to survey the huge number of galaxies in the universe. More distant objects are particularly neglected since light conforms to the inverse square law, so it takes four times longer to measure the spectrum of an object that is twice as far away.

2.1 Colours

The amount of light we receive from an object A is its *intensity* I_A. Its units are power per area, but we usually measure it in a logarithmic scale called the *apparent magnitude*. The apparent magnitude m_A of an object A is measured relative to the apparent magnitude of Vega, defined to be 0. An 100-fold increase in the intensity of an object results in a magnitude that is lower by 5: for objects A and B,

$$m_A - m_B = -5 \log_{100} \left(\frac{I_A}{I_B} \right).$$

Counterintuitively, brighter objects have a lower magnitude. The sun has an apparent magnitude of -27 and is the brightest object in the sky.

We use filters to separate light coming from different sections of the visible light spectrum. In our datasets, these are u, g, r, i, and z[(1)], in increasing order of wavelength. The u band is ultraviolet, g and r are visible, and i and z are infrared. These five filters have a large spread, providing information about objects' emissions in a wide part of the spectrum.

We use pairwise differences of magnitudes in these bands as the *colours* in photometric redshifts. The colours are $m_u - m_g$, $m_u - m_r$, $m_r - m_i$, and $m_i - m_z$. [31] Since the magnitude is a logarithmic scale, for bands a and b, $m_a - m_b$ is proportional to I_b / I_a, making the colours

[(1)] In astronomy, z can refer either to an object's redshift or to the infrared band. The meaning of any particular z should be clear from context.

independent of the apparent magnitude of the object as a whole. In addition to the four colours, we use m_r as a proxy for the apparent magnitude of the object.

2.2 Datasets

To predict photometric redshifts, we need a dataset of example objects. For each, we must know the five photometric bands, as well as the spectroscopic redshift. Uncertainty bounds for the photometric data and the redshift are very common, but not used in our model.

Many suitable datasets exist. The Sloan digital sky survey (SDSS) is very common, and contains a very large number of objects. The 2-degree Field Lensing Survey (2dFLenS) is another dataset we use, with far fewer objects. They have different distributions and cover different ranges of redshifts and magnitudes, permitting us to study the behaviour of our model under differing distributions of data.

2.2.1 Sloan digital sky survey

The Sloan digital sky survey (SDSS) is a large photometric and spectroscopic survey of the Northern sky. As of its twelfth data release, it contains useful spectroscopic redshifts for 4 266 444 objects, including 2 401 952 galaxies, along with the corresponding photometric measurements. [1] Of these galaxies, we select those with a redshift uncertainty of less than 0.1, leaving us with 1 707 233 examples.

The spectroscopic redshift distribution of objects in SDSS is presented in fig. 2.1. We see that it covers redshifts from about $z_{\text{spec}} = 0.05$ to about $z_{\text{spec}} = 0.8$, with some examples existing outside these bounds. The distribution of object magnitudes is shown in fig. 2.2. We see that SDSS covers r-band magnitudes from about 23 to about 16.

Figure 2.3 shows the distribution of SDSS in the spectroscopic redshift and the r-band magnitude. It is easily seen that these are quite correlated, with the relationship appearing linear.

2.2.2 The 2-degree field lensing survey

The 2-degree field lensing survey (2dFLenS) is also a photometric and spectroscopic survey. [34] Unlike SDSS, it observes the Southern sky. It lists 50 919 examples with spectroscopic redshifts and photometric measurements, which is significantly fewer than SDSS.

The spectroscopic redshift distribution of objects in 2dFLenS is presented in fig. 2.1. Its data spans from about from about $z_{\text{spec}} = 0.05$ to about $z_{\text{spec}} = 0.3$, with limited examples existing outside this interval. This is a lower spread than SDSS. As seen in fig. 2.2, 2dFLenS covers r magnitudes from about 19.5 to about 14. It has a significantly lower proportion of faint objects than SDSS.

We show a distribution of 2dFLenS on the plane of spectroscopic redshift and r-band magnitude in fig. 2.3. We see that, albeit a correlation exists, it appears much weaker than in SDSS. The data appears much more spread out.

Unlike SDSS, 2dFLenS also contains the infrared $W1$ and $W2$ from the Wide-field infrared survey explorer [35]. Its photometric redshifts add the colours $m_r - m_{W1}$ and $m_{W1} - m_{W2}$. [34] We do not use these for consistency with SDSS.

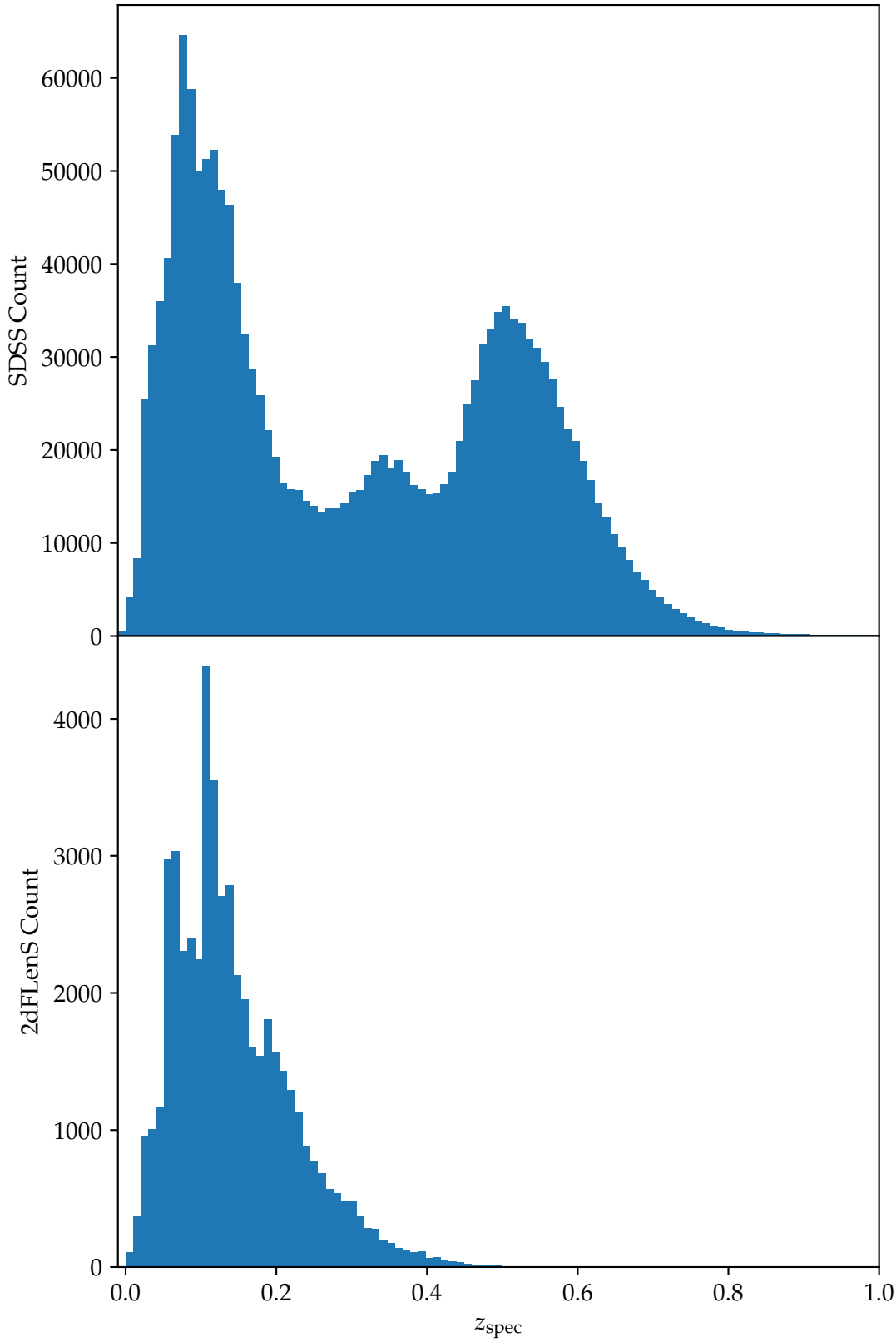


Figure 2.1: Histogram of objects in SDSS and 2dFLenS by spectroscopic redshift.

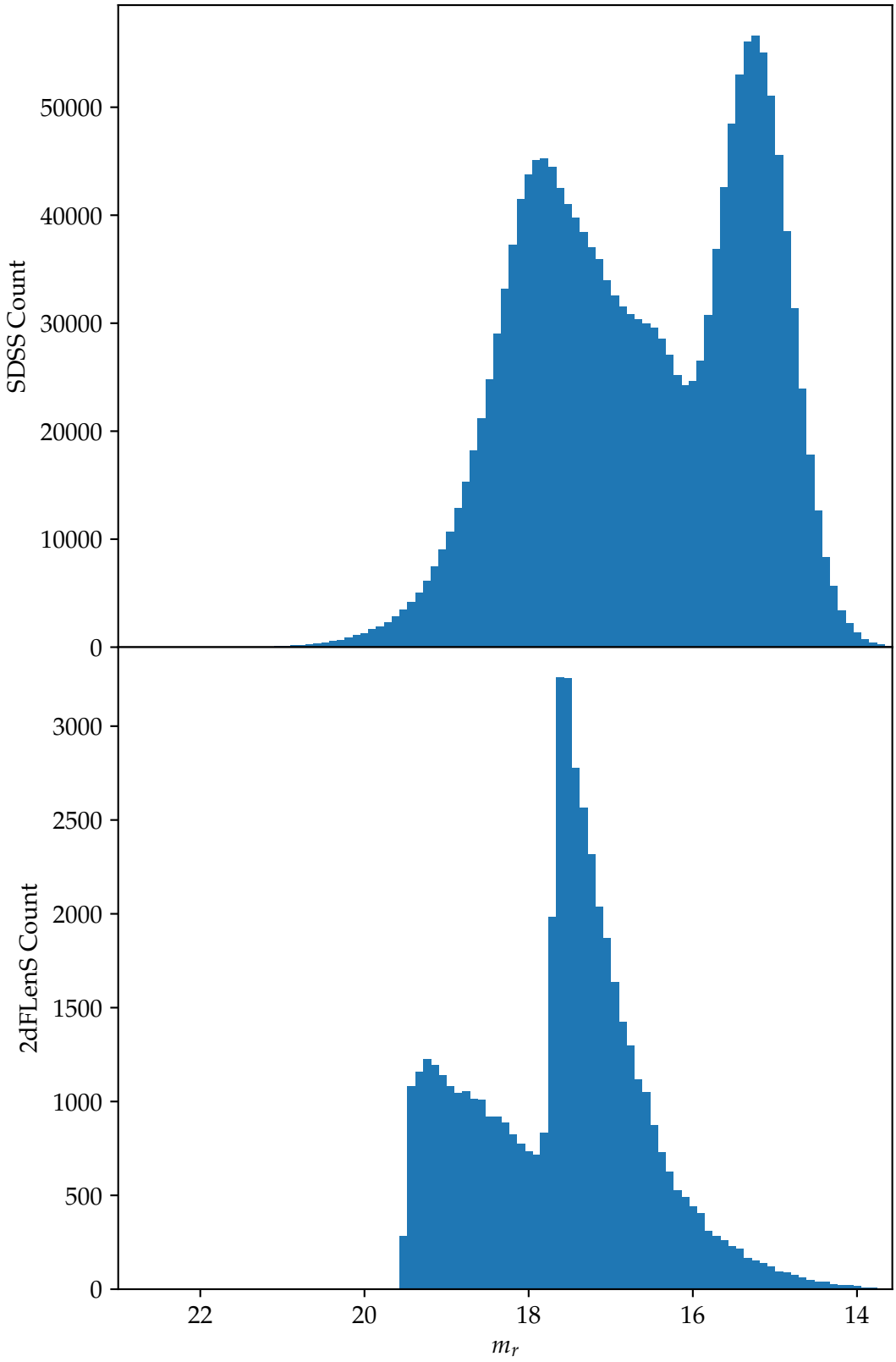


Figure 2.2: Histogram of objects in SDSS and 2dFLenS by magnitude in the r band.

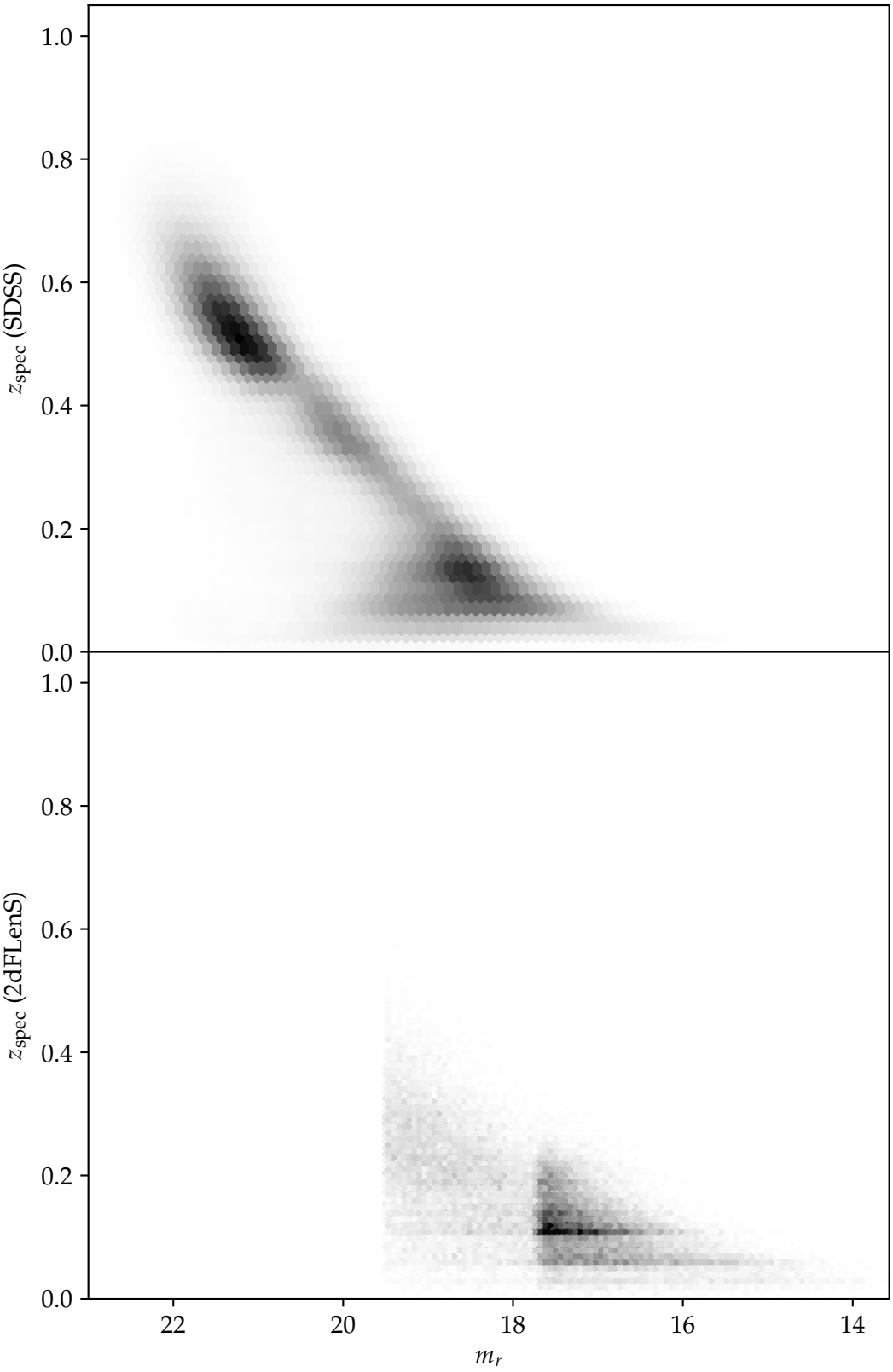


Figure 2.3: Distribution of SDSS and 2dFLenS by spectroscopic redshift and r-band magnitude.

Chapter 3

Regression on photometric redshifts

Given some spectroscopic labels, we wish to find redshifts from photometric data. This problem can be modelled using regression. We introduce regression and develop a predictor for photo-z and report its performance. We then extend regression to Gaussian processes, a family of models capable of reporting their confidence in their own predictions. These are later used in active learning to lower the number of spectra required to further improve our accuracy.

To formalise regression, let $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$ both be bounded. Let f map $\mathcal{X}$ to $\mathcal{Y}$. We are given a set

$$\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$$

of *n samples* of f, where $y_1, \ldots, y_n$ are the *labels*. Let h be the probability distribution of the label noise. Then for all i,

$$y_i = f(\mathbf{x}_i) + \epsilon \text{ for some error } \epsilon \sim h.$$

Regression is the task of reconstructing f from the finitely many examples of its inputs and outputs we have. $\mathcal{D}$ is the *training set*. Define $\mathfrak{D} \subset \mathcal{P}(\mathcal{X} \times \mathcal{Y})$ to be the space of possible training sets.

In our case, $\mathcal{X}$ is the space of photometric measurements that contains our input samples, and $\mathcal{Y} \subset (-1, \infty)$ is the space of redshifts spanned by our labels. Let f is the physical function that maps photometric measurements to redshifts. Since it is physically possible to have two objects with the same photometry but different redshifts, it is not obvious why it is valid to assume that such a function exists. To resolve this, we let f be the function that maps photometric measurements to to any corresponding object's *most likely* redshift. This lets us treat any inconsistency as any other source of noise.

To solve a regression problem, let $\hat{\mathcal{F}}$ be our space of approximations that we consider; its exact membership depends on our assumptions. A function $F : \mathfrak{D} \to \hat{\mathcal{F}}$ is our *model*. Given a training set, it returns a predictor; this is referred to as *training* the model. Define $\hat{f} \coloneqq F(\mathcal{D})$ to be our predictor.

We want to find $\hat{f} \in \hat{\mathcal{F}}$ that is the best approximation of f. Each candidate approximation $\hat{f}_c \in \hat{\mathcal{F}}$ can be assigned a likelihood that quantifies how good an approximation it is:

$$p\left(\hat{f}_c \,\middle|\, \mathcal{D}\right).$$

We define $\hat{f}$ to be the function that maximises this likelihood:

$$\hat{f} = F(\mathcal{D}) \coloneqq \operatorname*{argmax}_{\hat{f}_c \in \mathcal{F}} p\left(\hat{f}_c \,\middle|\, \mathcal{D}\right).$$

The definition of this likelihood depends on our assumptions about f, h, and F.

In our specific case we always assume that the noise is distributed according to a Gaussian with a 0 mean and a with variance σ^2 that we leave as a hyperparameter[1].

Some models are capable of not only predicting f but also returning a measure of their confidence in their own prediction. Gaussian process regression (GPR) is an example of such a model. This measure of confidence becomes crucial when we seek to improve our predictions with active learning.

3.1 Assessing performance

To assess the performance of a model, we must have a way of quantifying its accuracy. We withhold a portion of our data into a *testing set* $\mathcal{D}' \in \mathfrak{D}$ that is disjoint from $\mathcal{D}$. Let $n' := |\mathcal{D}'|$ and index the elements

$$\mathcal{D}' = \{(\mathbf{x}'_1, y'_1), \ldots, (\mathbf{x}'_{n'}, y'_{n'})\}$$

.

For each $i = 1, \ldots, n'$ define the prediction as

$$\hat{y}'_i := \hat{f}(\mathbf{x}'_i).$$

Define the δz error as

$$\delta z_i := \hat{y}'_i - y'_i,$$

as used in [17] and [34]. For redshifts, define dz as

$$dz_i := \frac{\delta_i}{1 + y'_i},$$

as in [34]. This accounts for the physical significance of the redshift: $1 + z$ is the ratio of the size of the universe when the light was emitted to its size now. Then dz_i is proportional to the resulting error when we use our predictions to calculate the historical size of the universe.

For a perfect predictor,

$$\hat{y}'_i = y'_i \text{ for all } i = 1, \ldots, n',$$

so $\delta_i = 0$. Unfortunately, perfect predictors are rarely possible to produce. This is due to our fundamental ability to reconstruct an arbitrary function from a finite number of samples[2]. Even knowing the form of f, noise label will reduce the accuracy of our predictor.

There exist many scores that permit us to compare predictors. The fraction of variance unexplained is a commonly used score. Others, more specific to photometric redshifts, are the mean dz error, the dz deviation, and the fraction of dz outliers. Photometric redshifts suffer from a high number of outliers; their influence is reduced with these redshift-specific scores.

3.1.1 Fraction of variance unexplained

The fraction of variance unexplained (FVU) measures the fraction of the variance of the labels $y'_1, \ldots, y'_{n'}$ not explained by the predictor.

To define it, let $\mathrm{VAR}_{\mathrm{err}} := \frac{1}{n'} \sum_{i=1}^{n'} \delta_i^2$ be the mean squared δ error. If the mean δ error is zero, this equals the variance[3] of the δ errors. Let $\mathrm{VAR}_{\mathrm{tot}} := \mathrm{var}\{y'_1, \ldots, y'_{n'}\}$ be the variance of the

[1] A hyperparameter is a parameter that depends on the distribution of our data.

[2] The space $\mathcal{F}$ of functions $\mathcal{X} \to \mathcal{Y}$ has a higher cardinality than the space $\mathfrak{D}$ of training sets. Any given model F is thus not bijective. Therefore there exists a function $f \in \mathcal{F}$ that cannot be approximated by F from a finite dataset in $\mathfrak{D}$.

[3] For a finite set $S = \{s_1, \ldots, s_{|S|}\} \subset \mathbb{R}$ of $|S|$ elements, we define the variance as $\mathrm{var}\, S := \frac{1}{|S|} \sum_{i=1}^{|S|} (s_i - \bar{s})^2$, where $\bar{s} := \frac{1}{|S|} \sum_{i=1}^{|S|} s_i$ is the mean of S.

labels of the testing set. Note that $\mathrm{VAR}_{\mathrm{err}}$ varies with the predictor, whereas $\mathrm{VAR}_{\mathrm{tot}}$ is constant for each testing set. We define

$$\mathrm{FVU} := \frac{\mathrm{VAR}_{\mathrm{err}}}{\mathrm{VAR}_{\mathrm{tot}}}.$$

Values of FVU closer to 0 are better. Intuitively, a perfect predictor explains the entire variance of the labels, so we expect it to yield $\mathrm{FVU} = 0$. Indeed, a perfect predictor has $\delta_1 = \cdots = \delta_{n'} = 0$, so $\mathrm{VAR}_{\mathrm{err}} = 0$. For a predictor that always returns the mean of the testing set, $\mathrm{VAR}_{\mathrm{err}} = \mathrm{VAR}_{\mathrm{tot}}$ by definition, so $\mathrm{FVU} = 1$, as expected since a predictor that always returns the mean explains none of the variance. A predictor with $\mathrm{FVU} > 1$ then performs worse by our metric than a predictor that always returns the mean.

3.1.2 Mean dz error

The mean dz error is specific to photometric redshifts. It is defined as the mean of all absolute dz errors:

$$\text{mean } dz \text{ error} := \mathrm{avg}\{|dz_1|, \ldots, |dz_{n'}|\}.$$

Values closer to 0 are better.

Unlike the fraction of variance unexplained error, which computes the sum of squared errors, the mean dz error depends on a sum of absolute errors (it is a weighted ℓ_1 loss). This is an important distinction, as the sum of squared errors is more likely to be overly influenced by outliers.

3.1.3 dz bias

The dz bias is also specific to photometric redshifts and is used in [34]. It quantifies whether the predictor is more likely to over- or underestimate the redshift. It is defined as to be the mean of all signed dz errors:

$$dz \text{ bias} := \mathrm{avg}\{dz_1, \ldots, dz_{n'}\}.$$

Unlike the FVU and the mean dz error, the dz bias is permitted to be negative. In addition, a dz bias of zero, whilst desirable, does not indicate a perfect predictor.

3.1.4 dz deviation

The dz deviation is defined as the half-width of the deviation of dz containing 68.3% of all objects. It is used in [34]. More formally, the dz deviation is the smallest $\sigma \in \mathbb{R}_{\geq 0}$ such that 68.3% of dz_i errors are within $[-\sigma, \sigma]$. It has the advantage of completely discarding outliers beyond percentile 68.3, vastly reducing their influence.

3.1.5 Fraction of dz outliers

The fraction of dz outliers is defined by [34] as the fraction of objects with $|dz| > 0.1$. It quantifies the fraction of objects whose redshifts fail to be explained by the model.

3.2 Existing methods

A number of methods are currently used for prediction of photometric redshifts. These can be split between empirical methods and template methods. Empirical methods are the sort that we are used to in machine learning: there exists a training set and we fit to it. Template methods rely on our modelling of an object's spectrum, and are much less accurate. Empirical methods can be further split into generative models that provide more flexibility and hence better accuracy, and discriminative models that are much faster.

3.2.1 Kernel density estimation

The kernel density estimation (KDE) used in photometric redshifts is a generative model that uses a six-dimensional Euclidean space, with five dimensions for our features and one dimension for our labels. For each sample in our training set, we insert a Gaussian aligned with the axes. The Gaussian has a mean equalling the features and label of the data. The standard deviation in each axis corresponds to the uncertainty of the corresponding measurement. The sum of these Gaussian yields a probability distribution over our six-dimensional space.

For a new sample, we construct a five-dimensional Gaussian with a mean equal to our features. Its standard deviation in each axis corresponds to the uncertainty of our measurements. Convolving it along the label axis, we find a probability distribution for the redshifts. We use it to compute our mean and uncertainty. [34]

KDE achieves an excellent fraction of dz outliers error of 0.0296 at the cost of a slow running speed.

3.2.2 Neural networks

A feed-forward neural network can be thought of as a nested series of linear regressors whose outputs are transformed using a non-linear *activation function*. [9] They can be used for photometric redshift prediction. [11] They work very well on datasets with abundant training examples, achieving a fraction of dz outliers error of 0.0354 on 2dFLenS, which is inferior to KDE. [34] Neural networks' discriminative nature makes them faster than KDE.

Despite their good accuracy, neural networks are problematic for photometric redshifts as their architecture makes it difficult to obtain confidence intervals on their predictions. Probabilistic neural networks [27] can be used to output a probability density function.

3.2.3 Boosted decision trees

A decision tree is a classifier that works by recursively splitting our feature space parallel to the axes based on individual features. To use it, we must first split our label space into a series of discrete classes. Boosting re-weights previously misclassified objects, and trains a new tree. [12] A boosted decision tree is an ensemble of these trees, where each member's vote is weighted according to their misclassification rate.

Boosted decision trees achieve an fraction of dz outliers error of 0.0315 on 2dFLenS. [27] This is superior to neural networks, but inferior to KDE. They are, however, much faster than KDE.

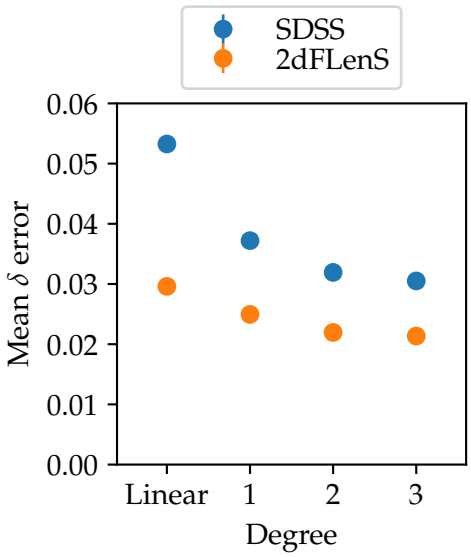
3.2.4 Template methods

Template methods are unlike the empirical methods introduced so far. They do not rely on fitting to a training set. Instead, they model objects' spectral energy distributions–the map from the wavelength to its flux density–using templates generated from observations and simulations. After accounting for redshift and correcting for intergalactic extinction, the distributions are compared to objects' colours to determine the redshift that fits best. [7]

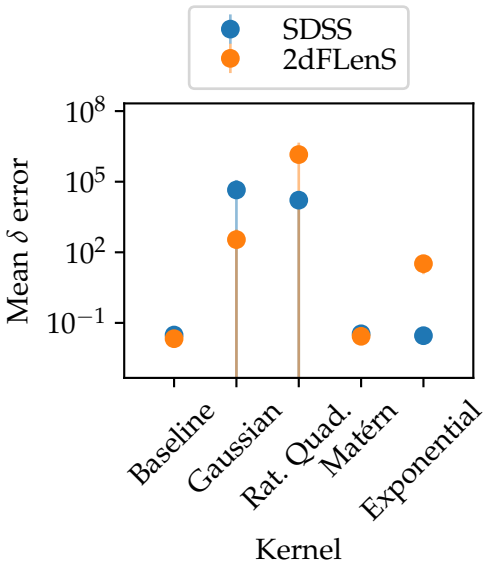
Template methods suffer from high biases and large errors on the 2dFLenS dataset, as shown in [34], where they achieve a fraction of dz outliers error of 0.0566. They are significantly outperformed by empirical methods.

3.3 Linear regression

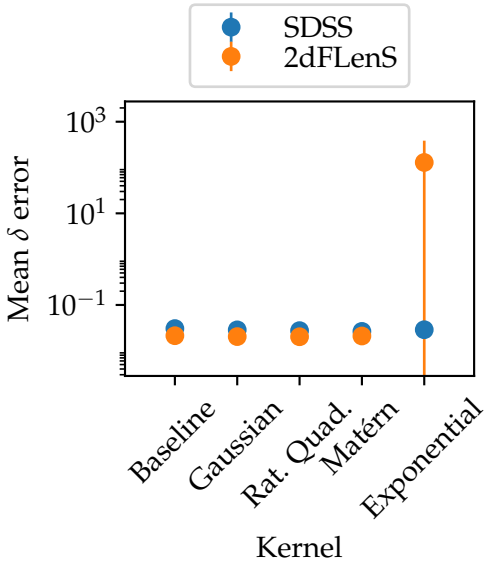
Linear regression is a simple regression model that expresses $\hat{f}(\mathbf{x})$ as a fixed linear combination of the components of $\mathbf{x}$. [9] We derive it and measure its performance.



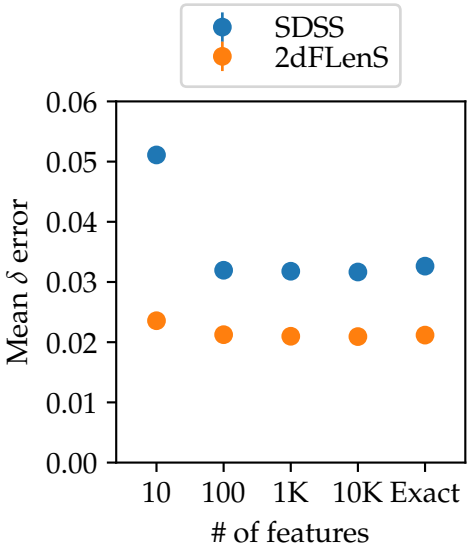
(a) Error of linear regression with polynomial features. Note that degree 1 polynomial features are linear regression with a bias term.



(b) Error of kernelised linear regression. We set σ to be the median distance between training $\mathbf{x}$ values. Baseline is regression with linear features.



(c) Error of regularised kernelised linear regression. The regularisation constant λ is 0.001. Baseline is unregularised regression with linear features.



(d) Error of linear regression using the RBF sampler approximation to the Gaussian kernel by the number of features.

Figure 3.1: Performance of different models tested. Each model was trained on 10 000 random data points. The error bars indicate a one-sigma confidence interval.

More formally, we suppose that

$$\hat{f}(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}.$$

We assume that $p(\hat{f} \mid \mathcal{D})$ depends only on how well $\hat{f}$ approximates f at the training samples. Assuming Gaussian likelihoods with variance σ^2, we wish to find $\mathbf{w}$ that yields the best predictor. We derive the least squares error.

By our assumption of Gaussian likelihoods,

$$\begin{aligned} p\left(\hat{f} \;\middle|\; \mathcal{D}\right) &= \prod_{i=1}^{n} \mathcal{N}\left(\hat{y}_i \mid y_i, \sigma^2\right) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{\tau\sigma^2}} \exp\left(-\frac{(\hat{y}_i - y_i)^2}{2\sigma^2}\right). \end{aligned} \tag{3.1}$$

We wish to find $\mathbf{w}$ that maximises this likelihood. This is equivalent to minimising the negative log likelihood:

$$\begin{aligned} -\log p\left(\hat{f} \;\middle|\; \mathcal{D}\right) &= -\sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{\tau\sigma^2}} \exp\left(-\frac{(\hat{y}_i - y_i)^2}{2\sigma^2}\right)\right] \\ &= -\sum_{i=1}^{n} \left(-\frac{1}{2}\log\left(\tau\sigma^2\right) - \frac{(\hat{y}_i - y_i)^2}{2\sigma^2}\right) \\ &= \frac{n}{2}\log\left(\tau\sigma^2\right) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2. \end{aligned}$$

Since $\frac{n}{2}\log(\tau\sigma^2)$ and $1/\sigma^2$ are constant, minimising the negative log likelihood is equivalent to minimising $\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$. We call this our *loss* function and write

$$E(\mathbf{w}) \coloneqq \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2.$$

This is called the *sum of squares* loss. Note that we can compute it without knowing σ^2.

minimising $-\log p(\mathbf{y} \mid \hat{f}, X)$ is equivalent to minimising $\frac{1}{2\sigma^2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$. Further, since $1/\sigma^2$ is constant, this is equivalent to minimising $\frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$.

The optimal weights vector is given by

$$\operatorname*{argmax}_{\mathbf{w}\in\mathbb{R}^d} p\left(\hat{f} \;\middle|\; \mathcal{D}\right) = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^d} E(\mathbf{w}).$$

We can solve this analytically. We rewrite

$$E(\mathbf{w}) = \frac{1}{2}(X\mathbf{w} - y)^\top (X\mathbf{w} - y),$$

where $X \in \mathbb{R}^{n\times d}$ such that $X_i \coloneqq \mathbf{x}_i$. Setting the derivative to zero to find the minimum,

$$\frac{dE(\mathbf{w})}{d\mathbf{w}} = X^\top(X\mathbf{w} - \mathbf{y}) = \mathbf{0}.$$

The minimum is

$$\mathbf{w} = \left(X^\top X\right)^{-1} X^\top \mathbf{y}.$$

We show the performance of purely linear regression in fig. 3.1a. It performs rather poorly, prompting us to consider non-linear features.

3.3.1 Feature differences

We show that creating new features that are pairwise differences between existing features does not affect a linear model since the same predictor will be found using the original features. This means that, for a purely linear model, our feature space of photometric colours is equivalent to that of photometric magnitudes.

Let $\mathbf{x} \in \mathcal{X}$ and let $G \in \mathbb{R}^{d \times d}$ be the matrix of pairwise differences:

$$G_{ij} \coloneqq x_i - x_j.$$

Let $\boldsymbol{\alpha} \in \mathbb{R}^d$ be a weights vector for $\mathbf{x}$ and let $\mathcal{B} \in \mathbb{R}^{d \times d}$ be a weights matrix for G. Define our predictor as

$$\hat{f}(\mathbf{x}) = \boldsymbol{\alpha} \cdot \mathbf{x} + \langle \mathcal{B}, G \rangle_{\mathrm{F}}, \tag{$*$}$$

where $\langle \cdot, \cdot \rangle_{\mathrm{F}}$ is the Frobenius inner product.

Let $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^d$ be defined by

$$\tilde{\boldsymbol{\alpha}}_i \coloneqq \boldsymbol{\alpha}_i + \sum_{j=1}^{n} \mathcal{B}_{ij} - \sum_{j=1}^{n} \mathcal{B}_{ji}.$$

We show that a predictor defined as

$$\tilde{f}(\mathbf{x}) = \tilde{\boldsymbol{\alpha}} \cdot \mathbf{x}$$

is equivalent to $(*)$.

We have that

$$\begin{aligned}
\tilde{\boldsymbol{\alpha}} \cdot \mathbf{x} &= \sum_{i=1}^{n} \mathbf{x}_i \left(\boldsymbol{\alpha}_i + \sum_{j=1}^{n} \mathcal{B}_{ij} - \sum_{j=1}^{n} \mathcal{B}_{ji} \right) \\
&= \sum_{i=1}^{n} \mathbf{x}_i \boldsymbol{\alpha}_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{x}_i \mathcal{B}_{ij} - \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{x}_i \mathcal{B}_{ji} \\
&= \sum_{i=1}^{n} \mathbf{x}_i \boldsymbol{\alpha}_i + \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{x}_i \mathcal{B}_{ij} - \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{x}_j \mathcal{B}_{ij} \\
&= \boldsymbol{\alpha} \cdot \mathbf{x} + \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{B}_{ij} \left(\mathbf{x}_i - \mathbf{x}_j \right) \\
&= \boldsymbol{\alpha} \cdot \mathbf{x} + \langle \mathcal{B}, G \rangle_{\mathrm{F}}
\end{aligned}$$

as desired.

Note that this does not apply to models where a nonlinear transformation is applied to G. It also does not necessarily hold when regularisation is applied to the weights.

3.3.2 Nonlinear features

The constraint that $\hat{y}$ must be a linear combination of the elements of $\mathbf{x}$ is very limiting. Few functions are well approximated by a line. To generalise linear regression, hence allowing us to approximate a wider range of functions, we can choose some basis functions $\phi_1, \ldots, \phi_{d'}$, each mapping $\mathbb{R}^d \to \mathbb{R}$. Let $\boldsymbol{\phi}$ be the d'-dimensional vector of these functions. For notational convenience, we assume that $\boldsymbol{\phi}$ refers to $\boldsymbol{\phi}(\mathbf{x})$ and $\Phi \in \mathbb{R}^{n \times d'}$ is the matrix of basis functions for every point in the training set.

To derive our linear model, we assume that $\hat{f}(\mathbf{x}) = \boldsymbol{\phi}^\top \mathbf{w}$ for some $\mathbf{w} \in \mathbb{R}^{d'}$. Following a procedure analogous to linear regression, we find that the optimal solution lies at

$$\mathbf{w} = \left(\Phi^T \Phi \right)^\top \Phi^\top \mathbf{y}.$$

We apply non-linear features to a regression model in fig. 3.1a. In our case, models with feature dimensionality perform better. To increase the dimensionality further, and with it the performance, we try applying kernels.

3.4 Kernels

A kernel is a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that

$$k(\mathbf{x}, \mathbf{x}') = \langle \boldsymbol{\varphi}(\mathbf{x}), \boldsymbol{\varphi}(\mathbf{x}') \rangle,$$

where $\boldsymbol{\varphi} : \mathbb{R}^d \to H$ maps our feature space $\mathbb{R}^d$ to a real Hilbert space H.[16] Intuitively, we can think of a kernel as an inner product in a different space.

Kernels permit us to generalise our earlier basis transformations $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^m$ to $\boldsymbol{\varphi} : \mathbb{R}^d \to H$. Due to the possibly infinite-dimensional nature of H, we cannot use $\boldsymbol{\varphi}$ directly. Instead, we can take advantage of it by computing k.

Like other inner products, a kernel can be thought of as a similarity measure. For example, $k(\mathbf{x}, \mathbf{x}') = 0$, then $\mathbf{x}$ and $\mathbf{x}'$ are orthogonal in H, and thus not similar at all by our metric.

For notational convenience, define $K \in \mathbb{R}^{n \times n}$ such that

$$K_{ij} \coloneqq k(\mathbf{x}_i, \mathbf{x}_j).$$

For a given $\mathbf{x}$ to predict, define $K_* \in \mathbb{R}^n$ as

$$K_{*i} \coloneqq k(\mathbf{x}, \mathbf{x}_i),$$

and define $K_{**} \coloneqq k(\mathbf{x}, \mathbf{x})$.

3.4.1 Examples of kernels

Gaussian kernel

The Gaussian kernel, also known as the radial basis function (RBF) kernel, is very common. Its smoothness is very desirable for many applications and its shape resembles distributions of much real-world data. It is defined by

$$k(\mathbf{x}, \mathbf{x}') \coloneqq \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right),$$

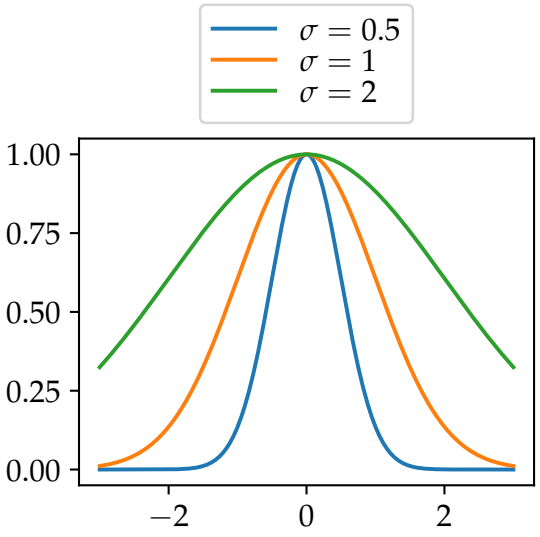
where the hyperparameter σ is known as the *length scale*. This is visualised in fig. 3.2a.

An more general definition involves d separate length scales $\sigma_1, \ldots, \sigma_d$, permitting us to set the length independently for each dimension. This definition is

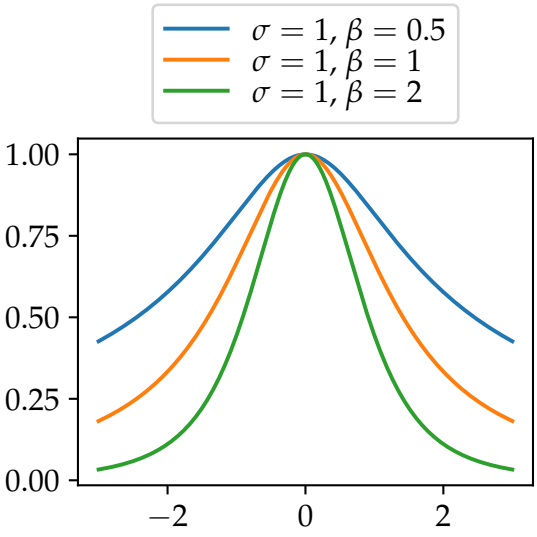
$$k(\mathbf{x}, \mathbf{x}') \coloneqq \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\top \Sigma^{-1} (\mathbf{x} - \mathbf{x}')\right),$$

where

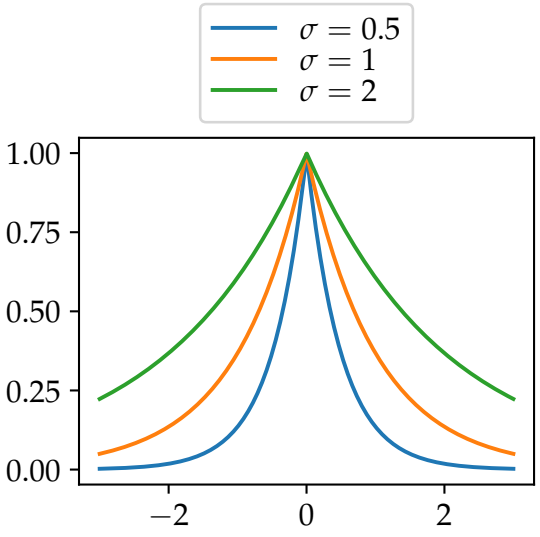
$$\Sigma = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_d^2 \end{bmatrix}.$$



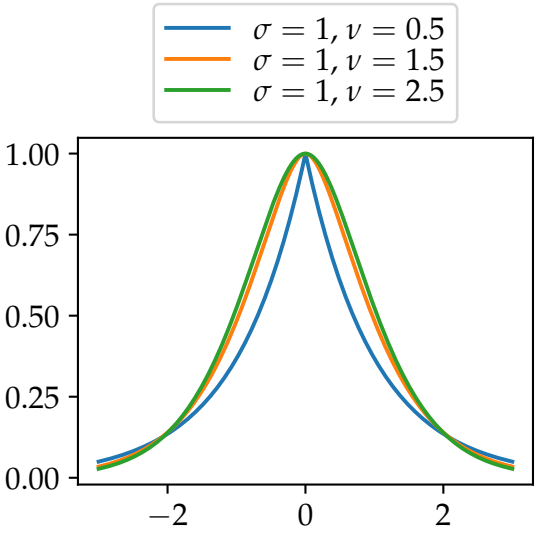
(a) $k(0, x)$, where k is the Gaussian kernel.



(b) $k(0, x)$, where k is the Rational-Quadratic kernel.



(c) $k(0, x)$, where k is the Exponential kernel.



(d) $k(0, x)$, where k is the Matérn kernel.

Figure 3.2: Plots of kernels.

Rational-quadratic kernel

The Rational-quadratic kernel is defined by

$$k(\mathbf{x}, \mathbf{x}') := \left(1 + \frac{1}{2\sigma} \left\|\mathbf{x} - \mathbf{x}'\right\|^2\right)^{-\beta},$$

where $\sigma > 0$ is the *length scale* and $\beta > 0$ is the *shape parameter*. It is a reasonable approximation of the Gaussian kernel that does not require an exponentiation. It is shown in fig. 3.2b.

Setting $\sigma = \beta\sigma_0$ and limiting $\beta \to \infty$ yields a Gaussian kernel with length scale σ_0.

Exponential kernel

The exponential kernel is less common as it is not differentiable, limiting its usefulness to many models. Its definition is

$$k(\mathbf{x}, \mathbf{x}') := \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|}{\sigma}\right),$$

where $\sigma > 0$. It is plotted in fig. 3.2c.

Matérn kernel

For some $\mathbf{x}$ and $\mathbf{x}'$, define $d := \|\mathbf{x} - \mathbf{x}'\|$. The Matérn Kernel is defined by

$$k(\mathbf{x}, \mathbf{x}') := \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}d}{\sigma}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}d}{\sigma}\right)$$

K_ν is the modified Bessel function of the second kind, Γ is the Gamma function, $\nu > 0$ is a hyperparameter, and $\sigma > 0$ is the length scale. It is differentiable $\lfloor \nu - 1 \rfloor$ times. See fig. 3.2d for a visualisation.

For some values of ν it can be simplified. When $\nu = 1/2$,

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{d}{\sigma}\right);$$

when $\nu = 3/2$,

$$k(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{3}d}{\sigma}\right) \exp\left(-\frac{\sqrt{3}d}{\sigma}\right);$$

and when $\nu = 5/2$,

$$k(\mathbf{x}, \mathbf{x}') = \left(1 + \frac{\sqrt{5}d}{\sigma} + \frac{5d^2}{3\sigma^2}\right) \exp\left(-\frac{\sqrt{5}d}{\sigma}\right).$$

For $\nu = 1/2$, this kernel the exponential kernel. The other two expressions are an exponential kernel multiplied by a polynomial in d/σ.

3.4.2 Kernelised linear regression

To extend our range of possible feature maps, we rewrite linear regression to use kernels. Let $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$ be our feature map.

Recall that the loss function $E : \mathbb{R}^{d'} \to \mathbb{R}_{\geq 0}$ that we wish to minimise is

$$E(\mathbf{w}) = \frac{1}{2} \|\mathbf{y} - \Phi\mathbf{w}\|_2^2.$$

We know that $\mathbf{w} \in \text{span}\{\phi_1, \ldots, \phi_n\}$. Hence, there exists $\mathbf{v} \in \mathbb{R}^n$ such that

$$\mathbf{w} = \Phi^\top \mathbf{v}.$$

We can rewrite the error function as $E : \mathbb{R}^{d'} \to \mathbb{R}_{\geq 0}$ such that

$$E(\mathbf{v}) = \frac{1}{2} \left\| \mathbf{y} - \Phi\Phi^\top \mathbf{v} \right\|_2^2.$$

Letting $K \coloneqq \Phi\Phi^\top$ be the kernel matrix,

$$E(\mathbf{v}) = \frac{1}{2} \left\| \mathbf{y} - K\mathbf{v} \right\|_2^2.$$

Differentiating, we find

$$\frac{dE(\mathbf{v})}{d\mathbf{v}} = K(K\mathbf{v} - \mathbf{y})$$

using the fact that K is symmetric. The minimum exists at $E'(\mathbf{v}) = 0$. There may be multiple solutions. One solution is at

$$\mathbf{v} = K^{-1}\mathbf{y}.$$

For a given input $\mathbf{x}$, the prediction is then

$$\begin{aligned}
\hat{f}(\mathbf{x}) &= \phi^\top \mathbf{w} \\
&= \phi^\top \Phi \mathbf{v} \\
&= \phi^\top \Phi K^{-1} \mathbf{y} \\
&= K_*^\top K^{-1} \mathbf{y},
\end{aligned}$$

where we define

$$K_* \coloneqq \Phi^\top \phi.$$

We use kernelised linear regression in fig. 3.1b. We see that most of our kernel choices perform quite poorly, with the exception of the Matérn kernel. This is likely due to overfitting. We introduce regularisation to prevent it.

3.5 Regularisation

We often wish to ensure that $\hat{f}$ relatively simple to prevent overfitting. We can make the assumption that every element of the weights vector $\mathbf{w}$ is likely to be close to zero:

$$p(w_i) = \mathcal{N}(w_i \mid 0, \omega^2)$$

for some variance $\omega^2 > 0$.

We can then change the likelihood in eq. (3.1) to account for this:

$$\begin{aligned}
p\left(\hat{f} \,\middle|\, \mathcal{D}\right) &= \prod_{i=1}^{n} \mathcal{N}\left(\hat{y}_i \mid y_i, \sigma^2\right) \prod_{i=1}^{m} \mathcal{N}(w_i \mid 0, \omega^2) \\
&= \prod_{i=1}^{n} \frac{1}{\sqrt{\tau\sigma^2}} \exp\left(-\frac{(\hat{y}_i - y_i)^2}{2\sigma^2}\right) \prod_{i=1}^{m} \frac{1}{\sqrt{\tau\omega^2}} \exp\left(-\frac{w_i^2}{2\omega^2}\right).
\end{aligned}$$

Computing the negative log likelihood,

$$-\log p\left(\hat{f} \,\middle|\, \mathcal{D}\right) = \frac{n}{2}\log\left(\tau\sigma^2\right) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 + \frac{m}{2}\log\left(\tau\omega^2\right) + \frac{1}{2\omega^2}\sum_{i=1}^{m} w_i^2.$$

Eliminating the constant terms and defining $\lambda := \sigma^2/\omega^2$, we find that

$$\begin{aligned}
\underset{\mathbf{w}}{\operatorname{argmin}} \left[-\log p\left(\hat{f} \,\middle|\, \mathcal{D}\right)\right] &= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\frac{1}{2\sigma^2} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 + \frac{1}{2\omega^2} \sum_{i=1}^{m} w_i^2 \right) \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 + \frac{\lambda}{2} \sum_{i=1}^{m} w_i^2 \right) \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\frac{1}{2} (\hat{\mathbf{y}} - \mathbf{y})^\top (\hat{\mathbf{y}} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right) \\
&= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\frac{1}{2} (\Phi\mathbf{w} - \mathbf{y})^\top (\Phi\mathbf{w} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w} \right).
\end{aligned}$$

Our loss now depends on $\mathbf{w}$ and λ and is given by

$$E(\mathbf{w}, \lambda) = \frac{1}{2} (\Phi\mathbf{w} - \mathbf{y})^\top (\Phi\mathbf{w} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^\top \mathbf{w}, \tag{3.2}$$

where λ is our *regularisation constant*: a higher value implies a simpler model, reducing the likelihood of overfitting, but increasing the chance of underfitting.

We can find $\mathbf{w}$ that minimises the loss. Taking the derivative and setting it to $\mathbf{0}$,

$$\frac{dE(\mathbf{w}, \lambda)}{d\mathbf{w}} = \Phi^\top (\Phi\mathbf{w} - \mathbf{y}) + \lambda\mathbf{w} = \mathbf{0}.$$

Solving,

$$\mathbf{w} = \left(\Phi^\top \Phi + \lambda I\right)^{-1} \Phi^\top \mathbf{y}.$$

3.5.1 Regularisation for kernelised regression

To derive regularised kernelised linear regression, we follow a similar procedure. We apply the kernel trick to eq. (3.2), obtaining

$$E(\mathbf{v}, \lambda) = \frac{1}{2} (K\mathbf{v} - \mathbf{y})^\top (K\mathbf{v} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{v}^\top K\mathbf{v}.$$

Solving for $\mathbf{v}$,

$$\mathbf{v} = (K + \lambda I)^{-1} \mathbf{y}.$$

Then our predictor is

$$\hat{f}(\mathbf{x}) = K_*^T (K + \lambda I)^{-1} \mathbf{y}.$$

We performed regularised kernelised regression in fig. 3.1c. We see that albeit regularisation improves the performance of most kernels, regression on SDSS with the exponential kernel still underperforms. This is likely due to λ being too low, resulting in numerical errors. From these results, we settle on using the Gaussian kernel since it performs well. It is also convenient since it is easy to approximate.

3.6 Kernel approximations

For large dataset, computing K becomes expensive it as has a space complexity of $O(n^2)$. It is often desirable to approximate φ, the transformation to the kernel's Hilbert space. For such an approximation ϕ and its corresponding Φ, $K \approx \Phi\Phi^\top$. Using ϕ as our feature function permits us to take advantage of many of the benefits of kernels without the corresponding large matrices.

3.6.1 RBF sampling

RBF Sampling approximates the Gaussian (RBF) kernel by randomly sampling from its Fourier transform. We use the implementation from scikit-learn [19], which is a variant of random kitchen sinks [21]. It has the advantage of not requiring more hyperparameters than the kernel function itself. Despite its random nature, [21] derives a probabilistic lower bound on its accuracy.

We use RBF sampling in fig. 3.1d. The performance appears to converge to levels similar to exact on 100 candidate features. We settle on using a kernel approximation with 1000 features to permit us to obtain reliable uncertainties from Gaussian process regression.

3.6.2 Empirical kernel map

A kernel represents a sample by its similarity to all other training sample. To approximate it, we can represent each sample by its similarity to a subset of training samples.[24]

Let $\phi^n : \mathcal{D} \to \mathbb{R}^n$ such that

$$\phi^n(\mathbf{x}) \coloneqq (k(\mathbf{x}_1, \mathbf{x}), \ldots, k(\mathbf{x}_1, \mathbf{x}))^\top.$$

Let $\Phi^n \in \mathbb{R}^{n \times n}$ be such that

$$\Phi^n_i \coloneqq \phi_n(\mathbf{x}_i).$$

Then $\Phi^n(\Phi^n)^\top = K^2$.

Let $\tilde{K}^{-1/2} \in \mathbb{R}^{n' \times n}$ be a low-rank approximation of $K^{-1/2}$.[4] Define $\phi : \mathcal{X} \to \mathbb{R}^{n'}$ as

$$\phi \coloneqq \tilde{K}^{-1/2}\phi^n(\mathbf{x})$$

with a corresponding Φ. Then

$$\begin{aligned}\Phi\Phi^T &= \Phi^n \left(\tilde{K}^{-\frac{1}{2}}\right)^\top \tilde{K}^{-\frac{1}{2}} (\Phi^n)^\top \\ &\approx \Phi^n K^{-1} (\Phi^n)^\top \\ &= K.\end{aligned}$$

This approximation is often more accurate than RBF sampling since we may choose the subset of training samples used to approximate the data. However, choosing this subset is difficult, as is computing $\tilde{K}^{-1/2}$. This limits the utility of this approximation.

3.7 Finding hyperparameters

Our model is a function of six hyperparameters. We have the five length scales $\sigma_1, \ldots, \sigma_5$ for the Gaussian kernel, one for each dimension of our input space. We also have the regularisation constant λ. These must be tuned to avoid suboptimal performance. The default parameters in software frameworks are rarely suitable due to large differences between datasets. Guesswork is time-consuming and prone to failure.

We test three algorithms for finding hyperparameters. Grid search is the simplest model often outperformed by Bayesian optimisation, both in accuracy and speed. Automatic relevance determination occassionally outperforms the other methods, but in general it performs worse. The results are presented in table 3.1.

We settle on automatic relevance determination, although it is not the best performing method. This is because ARD is fast and the existing scikit-learn implementation is easy to use. ARD also has fewer hyperparameters itself.

[4] We want $(\tilde{K}^{-1/2})^T\tilde{K}^{-1/2} \approx K$.

	Defaults	Grid search	Bayesian opt.	ARD
FVU	0.092	0.079	0.078	0.090
Mean dz error	0.027	0.027	0.027	0.030
dz bias	-0.0015	-0.0014	-0.0012	-0.0021
dz deviation	0.029	0.029	0.028	0.032
Fraction of dz outliers	0.027	0.026	0.025	0.036

(a) Hyperparameter selection performance for SDSS.

	Defaults	Grid search	Bayesian opt.	ARD
FVU	0.18	0.17	0.18	0.18
Mean dz error	0.020	0.020	0.020	0.021
dz bias	-0.0010	-0.00089	-0.00088	-0.00087
dz deviation	0.024	0.024	0.024	0.025
Fraction of dz outliers	0.0062	0.0059	0.0062	0.0058

(b) Hyperparameter selection performance for 2dFLenS.

Table 3.1: Comparison of hyperparameter finding methods.

3.7.1 Grid search

Grid search is one technique that permits us to find hyperparameters. We must first define a finite set of candidate hyperparameter configurations H. We also withhold a *validation* set $\mathcal{D}'' \in \mathfrak{D}$ to measure the performance of our parameters. The algorithm, using scikit-learn [19] conventions, is:

```
def cross_validate(H, train_X, train_y, validation_X, validation_y)
    best_parameters = None
    best_score = float('-inf')

    for h in H:
        regressor.set_params(**h)
        regressor.train(train_X, train_y)
        score = regressor.score(validation_X, validation_y)

        if score > best_score:
            best_parameters = h

    return best_parameters
```

The name *grid search* comes from H often being a Cartesian product of the options for each hyperparameter, forming a multidimensional 'grid'.

Since grid search does not perform well in high-dimensional spaces, we limit the dimensionality of our search space by setting $\sigma := \sigma_1 = \cdots = \sigma_5$. We then try values of σ equalling the 10^{th}, 25^{th}, 50^{th}, 75^{th}, and 90^{th} percentiles for our data. We also try λ values of 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.1, and 1.

3.7.2 Bayesian optimisation

Our regression has six hyperparameters: a length scale σ_i for each of the five features and a regularisation constant λ. The high dimensionality of the hyperparameter space makes it difficult to find the optimal parameters using grid search. Bayesian optimisation may be used instead to speed up this process.

Bayesian optimisation globally minimises a black-box function without requiring derivatives. We place a prior on our space, and update the expected value of the function at a particular point after every evaluation. The algorithm suggests points where the function should be evaluated next to obtain the most useful information about our function.

Bayesian optimisation is used similarly to grid search, but we evaluate our performance at the suggested points instead of regular intervals to obtain the most information. This permits us to efficiently search large spaces.

We use the implementation in Hyperopt [8].

3.7.3 Automatic relevance determination

Automatic relevance determination (ARD) is another technique that may be used to find the hyperparameters for our GPR. [33] It solves a linear regression problem, together with an iterative algorithm for tuning its own regularisation.

Each of the five length scales $\sigma_1, \ldots, \sigma_5$ of our Gaussian kernel is inversely proportional to the precision of its corresponding feature. If the length scale is low, then a small change in the feature will have a large effect on the output. To find the precision of our five features and the regularisation constant, we use ARD.

In ARD, we apply an iterative algorithm to tune our regularisation. For our original regularisation, we find the variances $\varsigma_1^2, \ldots, \varsigma_5^2$ of the features. Letting Σ be a diagonal matrix such that $\Sigma_{ii} := \varsigma_i^2$. We let the loss of our regression be

$$E(\mathbf{w}) := \frac{1}{2} \|X\mathbf{w} - \mathbf{y}\|^2 + \lambda \mathbf{w}^\top \Sigma \mathbf{w}.$$

Intuitively, this is because the bigger the variance of a feature, the lower its weight should be to compensate.

ARD then repeatedly optimises the evidence of the model, as described in [14], to tune λ and Σ. Once these converge, we use them as the regularisation constant and length scales of our model.

We use the implementation in scikit-learn [19].

ARD does not obviously generalise to kernels with other parameters, such as the rational-quadratic kernel, which has a shape parameter. It also does not generalise to models that use different regularisation norms, such as ℓ_1.

3.8 Gaussian process regression

Gaussian process regression (GPR) is a predictor that yields confidence intervals for its own predictions. This is very useful when performing active learning. We derive GPR here.

A Gaussian process is a distribution over functions. Formally, a Gaussian process is defined as a 'collection of random variables, any finite number of which have a joint Gaussian distribution.'[22]

Defining

$$\begin{aligned} m(\mathbf{x}) &:= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &:= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned}$$

we write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

Then $\operatorname{cov}(\mathbf{y}) = K + \sigma_n^2 I$, where σ_n^2 is the variance of our label noise.

Under the prior, we have that the joint distribution of the observed targets and of the function values at the test locations is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}' \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K + \sigma_n^2 I & K_* \\ K_*^\top & K_{**} \end{bmatrix}\right).$$

It can then be derived that

$$\mathbf{y}' \mid X, \mathbf{y}, X' \sim \mathcal{N}\left(\hat{f}(X'), \operatorname{var} \hat{f}(X')\right),$$

where

$$\hat{f}(X) = K_*^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y},$$

and

$$\operatorname{cov} \hat{f}(X) = K_{**} - K_*^\top \left(K + \sigma_n^2 I\right)^{-1} K_*.$$

For a single example now, the value for which $\hat{f}(\mathbf{x})$ that has the highest probability is the mean of the Gaussian from which it is sampled. Hence, we can set

$$\hat{f}(\mathbf{x}) = K_*^\top \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y}, \tag{3.3}$$

with the variance of the prediction being

$$\operatorname{var} \hat{f}(\mathbf{x}) = K_{**} - K_*^\top \left(K + \sigma_n^2 I\right)^{-1} K_*.$$

3.8.1 Equivalence to kernelised linear regression

Observe that eq. (3.3) is the equation for kernelised linear regression with regularisation, where $\lambda = \sigma_n^2$. Thus, Gaussian Process regression and kernelised regular regression always return the same expected value, although kernelised linear regression is unable to estimate the variance of that prediction.

3.9 Approximating Gaussian processes

Gaussian process regression relies on our calculation of $(K + \sigma_n^2)^{-1}$. Matrix inversion has a large time complexity[5]. This does not scale well to large training sets. Instead, we can approximate GPR with algorithms of a lower time complexity.

3.9.1 Low-rank approximation

We can approximate this matrix product with a complexity that is linear in n. We can do so by approximating the kernel with a finitely-dimensional set of features, and reversing the kernelisation.

Let $\Phi \in \mathbb{R}^{n \times d'}$ for $d' \ll n$ be our kernel approximation for the training set, letting $K \approx \Phi\Phi^\top$. Let $\phi \in \mathbb{R}^{d'}$ be the kernel approximation for the value we wish to predict for, so $K_* \approx \Phi\phi$. We have that,

$$\begin{aligned} K_*^\top (K + \sigma_n^2)^{-1} &\approx \phi^\top \Phi^\top \left(\Phi\Phi^\top + \sigma_n^2 I\right)^{-1} \\ &= \phi^\top \Phi^\top \left(\sigma_n^{-2} I - \sigma_n^{-2} \Phi (\sigma_n^2 + \Phi^\top \Phi)^{-1} \Phi^\top\right) \\ &= \sigma_n^{-2} \phi^\top \Phi^\top - \sigma_n^{-2} \phi^\top \Phi^\top \Phi (\sigma_n^2 + \Phi^\top \Phi)^{-1} \Phi^\top, \end{aligned} \tag{3.4}$$

[5] Matrix inversion takes $O(n^3)$ time for a naive implementation and $O(n^{2.373})$ time for an optimised method.

where we apply the Woodbury matrix identity[20] to reduce the rank of the matrix being inverted.

Substituting eq. (3.4) into eq. (3.3) and section 3.8,

$$\hat{f}(\mathbf{x}) = \sigma_n^{-2}\boldsymbol{\phi}^\top\Phi^\top\mathbf{y} - \sigma_n^{-2}\boldsymbol{\phi}^\top\Phi^\top\Phi(\sigma_n^2 + \Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}$$

and

$$\operatorname{var}\hat{f}(\mathbf{x}) = \boldsymbol{\phi}^\top\boldsymbol{\phi} - \sigma_n^{-2}\boldsymbol{\phi}^\top\Phi^\top\Phi\boldsymbol{\phi} - \sigma_n^{-2}\boldsymbol{\phi}^\top\Phi^\top\Phi(\sigma_n^2 + \Phi^\top\Phi)^{-1}\Phi^\top\Phi\boldsymbol{\phi}.$$

Since the size of the matrix being inverted no longer depends on n, this is a linear-time approximation.

Numerical instabilities present difficulties for small σ_n^2 as the matrix $(\Phi\Phi^\top + \sigma_n^2 I)$ becomes closer to being singular.

Note that this formulation permits online learning in linear time as the matrix $\Phi^\top\Phi$ is the sum of outer products of our samples. To add more samples, we add the additional outer products to this matrix in linear time, and perform the inversion, which is constant time in the number of points we are adding.

3.9.2 Alternatives

The mean of GPR can also be approximated by linear regression. Letting Φ be the kernel approximation on the training set, we find that the regression weights are

$$\mathbf{w} = \left(\Phi^\top\Phi + \sigma_n^2 I\right)\Phi^\top\mathbf{y},$$

where σ_n^2 becomes the regularisation constant. This can be computed in $O(nd')$ time by solving the linear equation instead of directly performing the matrix inversion. It remains to find an approximation to the variance predicted by GPR. Kernel density estimation (KDE) can be used to approximate this. We expect the inverse density of the estimator to be correlated with the variance of the GPR. KDE density computation is generally much faster than GPR variance prediction, and more numerically stable than GPR approximations. KDE is used in [18] as a proxy for the uncertainty of a GPR. They use these uncertainties as the basis of an active learning recommender. This method was not used in this work, as we found GPR approximations sufficiently numerically stable.

There also exist sparse approximations of Gaussian process regression that can be trained in linear time. [6] [30] These are beyond our scope.

3.10 Result

We have thus chosen our predictor. We are using a low-rank approximation of Gaussian process regression with the Gaussian kernel. We approximate the kernel with RBF sampling with 1000 features. The regularisation constant and length scales for the kernel are found with automatic relevance determination.

We show the passive learning curves on SDSS and 2dFLenS for our predictor in fig. 3.3 and fig. 3.4. Since SDSS contains a large amount of data, we show a learning curve that reaches one million points in fig. 3.5. We see that even at very large numbers of training points, the accuracy of SDSS and 2dFLenS keeps improving. Much more data is needed to improve the accuracy of these models, making this problem an ideal candidate for active learning.

We can observe that SDSS performs better than 2dFLenS based on the FVU error, but 2dFLenS performs better according to the mean dz error. This is due to the spread of the data that we see in fig. 2.3. SDSS has a much stronger correlation between m_r and z, whereas the examples in 2dFLenS are much more spread out: there is a higher proportion of outliers. The FVU error places much more weight on these outliers than the mean dz error, leading to this discrepancy.

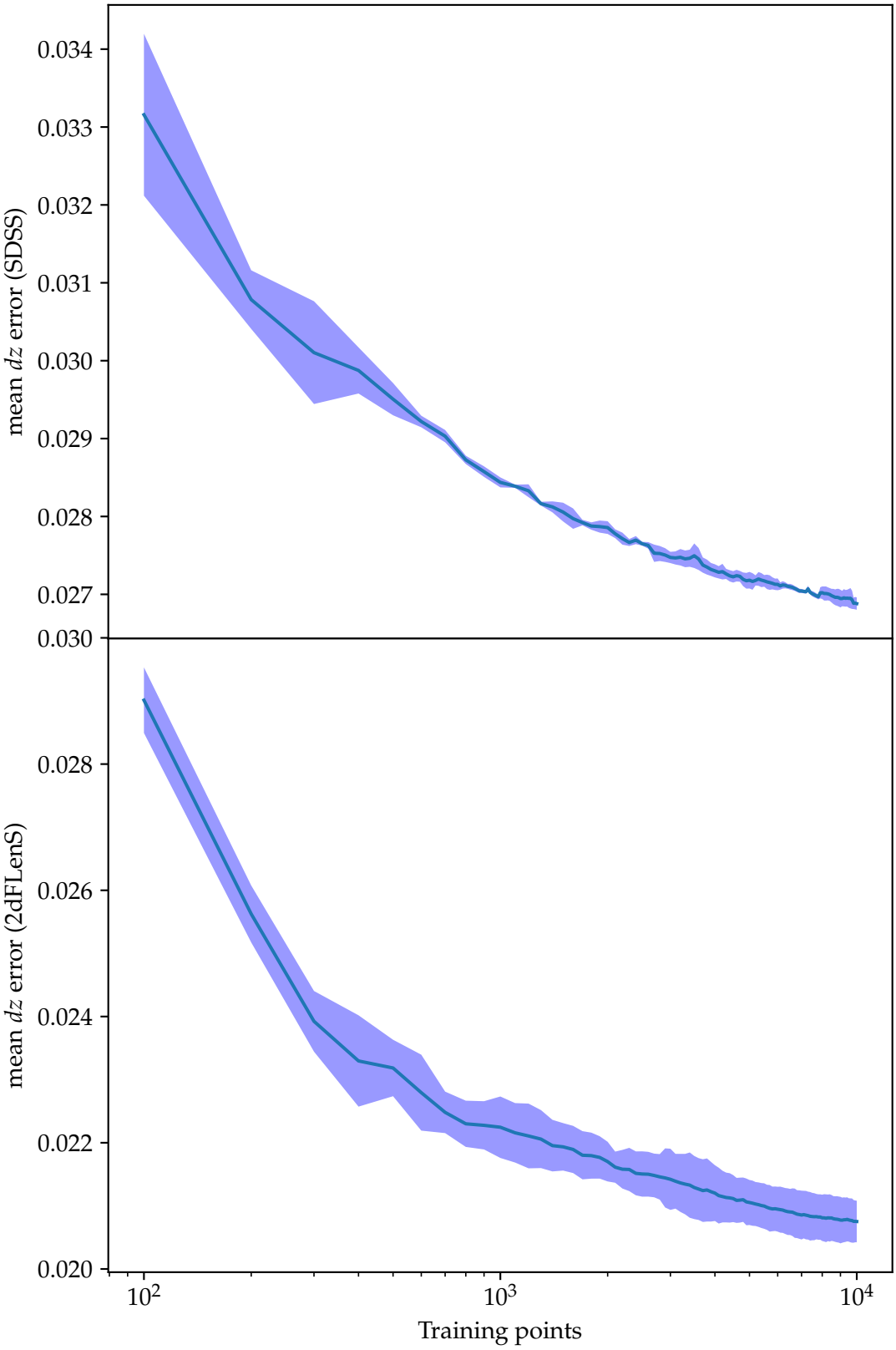


Figure 3.3: Passive learning curve: Mean dz error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen at random. The shaded interval represents the one-sigma confidence interval.

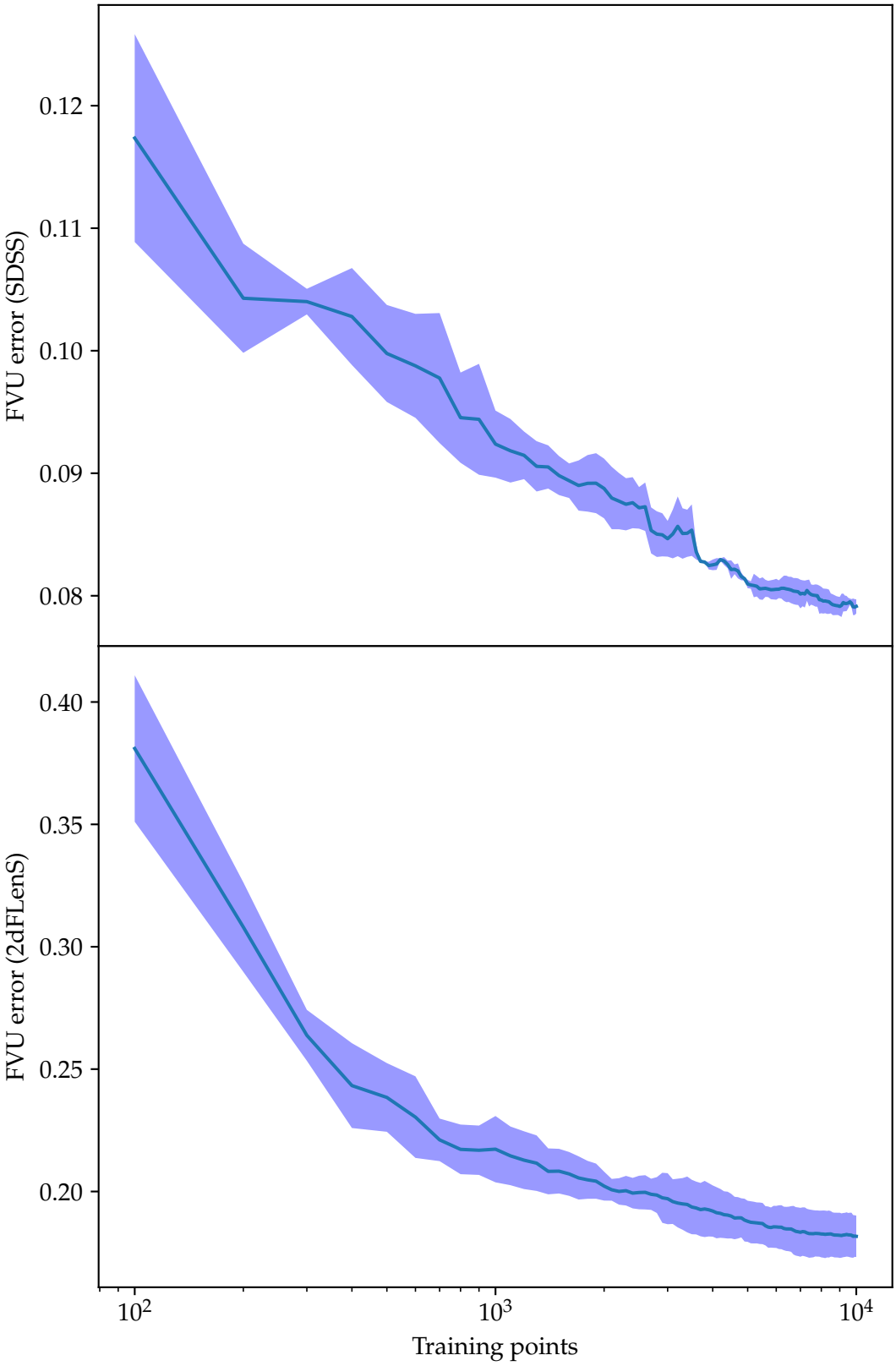


Figure 3.4: Passive learning curve: FVU error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen at random. The shaded interval represents the one-sigma confidence interval.

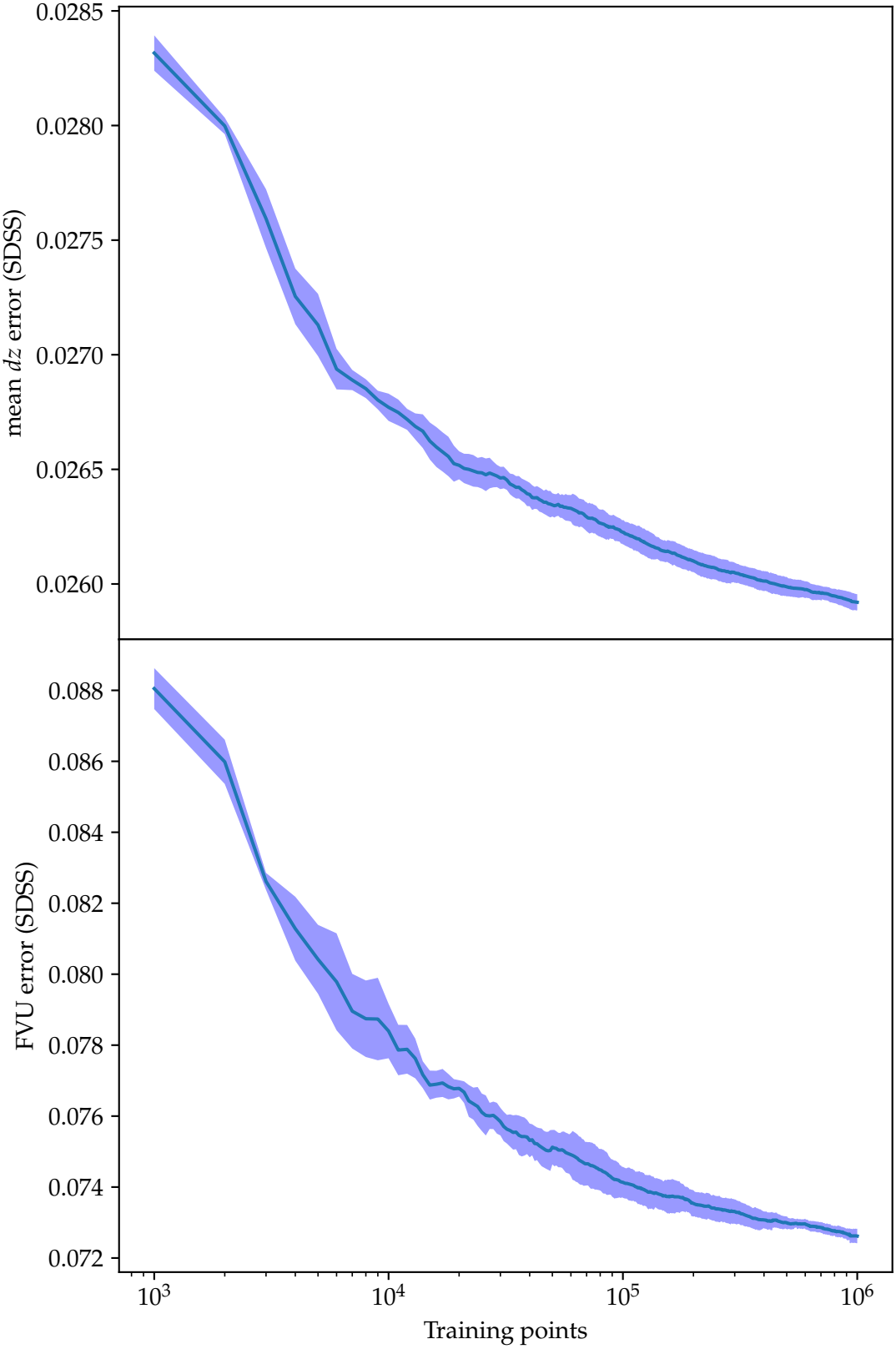


Figure 3.5: Long passive learning curve: Mean dz error and FVU error plotted against the number of training points for both SDSS, trained up to 1 million samples. The training points are chosen at random. The shaded interval represents the one-sigma confidence interval.

Chapter 4

Active learning

In regression on photometric redshifts, it is expensive to obtain new labels for training samples: a telescope can only measure the spectrum of a few galaxies at a time. Photometric data is cheap, so we have a large pool of unlabelled objects. We wish to spend our telescope time labelling the objects that are the most likely to improve the accuracy of our model. Active learning is a technique that permits this.

In active learning, we selectively find labels to samples based on their potential for improving our model. Given an existing model and a pool of unlabelled samples, a *recommender* chooses samples based on its policy. It is only after a sample is recommended that we query its label and add it to our model's training set. A procedure is presented in algorithm 1.

A recommendation strategy is effective if its choices yield a more accurate model than a randomly selected training set. Active learning then reduces, compared to the naïve approach, the number of labelled samples needed to achieve a predictor of a desired accuracy, saving resources.

4.1 Recommenders

In active learning, a *recommender* is a function[(1)] that tells us what unlabelled samples we should label next. It can be thought of in two equivalent ways.

A recommender can be thought of as a function $\pi : \mathcal{P}(\mathcal{X}) \times \mathbb{N} \to \mathcal{P}(\mathcal{X})$, where $\pi(P, r)$ returns the r most preferred samples from the sample pool P. A recommender can also be a function $\varpi : \mathcal{X} \to \mathbb{R}$. Then $\varpi(\mathbf{x})$ returns the preference score it assigns to $\mathbf{x}$, with higher scores being preferred. Define Π to be the space of all recommenders.

Given a recommender ϖ returning a score, we can define a recommender π returning a set of samples. One way to obtain $\pi(P, r)$ is by choosing the top r samples from P as scored by ϖ.

Similarly, given a recommender π returning a set of samples, we can define a recommender ϖ returning a score. Assuming a fixed pool $P \in \mathcal{P}(\mathcal{X})$, for all $\mathbf{x} \in \mathcal{X}$, define

$$\varpi(\mathbf{x}) \coloneqq -\min\{i = 1, \ldots, |P| \mid \mathbf{x} \in \pi(\mathcal{X}, i)\}.$$

Then the score ϖ assigns to $\mathbf{x}$ is the negation of its preference index in π.

4.2 Related areas

Bayesian optimisation on Gaussian process regression shares many concepts with active learning. In Bayesian optimisation we wish to find the minimum of a black-box function. We have a prior on where we think the minimum might lie. Gaussian processes permit us to model

[(1)] To be precise, a recommender is not always a function as it is permitted to be nondeterministic.

Algorithm 1: The active learning procedure.

Data:
A recommender $\pi : \mathcal{P}(\mathcal{X}) \times \mathbb{N} \to \mathcal{P}(\mathcal{X})$,
A function $\hat{F} : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{F}$ generating a predictor from training samples,
A training sample pool $P \subseteq \mathcal{X}$,
A labeller $\tilde{f} : P \to \mathcal{Y}$,
Step size $t \in \mathbb{Z}_{>0}$,
An error function $e \in \mathcal{E}$,
A desired maximum error $\text{err}_{\max}$.
Result: A predictor $\hat{f} : \mathcal{X} \to \mathcal{Y}$ with $e(\hat{f}) \leq e_{\max}$ or error
if $|P| < t$ **then return** error
$\mathcal{D}_{\mathcal{X}} \leftarrow t$ samples drawn randomly from P without replacement
$P \leftarrow P \setminus \mathcal{D}_{\mathcal{X}}$ (remove chosen samples from pool)
$\mathcal{D} \leftarrow \left\{ \left(\mathbf{x}, \tilde{f}(\mathbf{x})\right) \mid \mathbf{x} \in \mathcal{D}_{\mathcal{X}} \right\}$ (label chosen samples)
$\hat{f} \leftarrow F(\mathcal{D})$ (train model)
$\text{err} \leftarrow e\left(\hat{f}\right)$ (compute error)
while $\text{err} > \text{err}_{\max}$ **do**
  if $|P| < t$ **then return** error
  $\mathcal{D}_{\mathcal{X}} \leftarrow \pi(P, t)$ (get recommendations)
  $P \leftarrow P \setminus \mathcal{D}_{\mathcal{X}}$ (remove chosen samples from pool)
  $\mathcal{D} \leftarrow \mathcal{D} \cup \left\{ \left(\mathbf{x}, \tilde{f}(\mathbf{x})\right) \mid \mathbf{x} \in \mathcal{D}_{\mathcal{X}} \right\}$ (label chosen samples)
  $\hat{f} \leftarrow F(\mathcal{D})$ (train model on all samples)
  $\text{err} \leftarrow e\left(\hat{f}\right)$ (compute error)
end
return $\hat{f}$

the function and place uncertainty estimates on our predictions. Much like in active learning, we are permitted to query the function for its value at a given point, but we assume that this is expensive. Bayesian optimisation also uses recommenders in an attempt to maximise the value gained per query to our function. The recommenders in active learning and in Bayesian optimisation differ since the end goal of each algorithm is different. Bayesian optimisation recommenders may use the predicted value of the function, the uncertainty of the prediction, and the prior on the position of the minimum to make their recommendations. [26] [28] [29]

Optimal design is a technique that is similar to active learning, in that it wishes to minimise the number of labels needed to train a predictor. Optimal design is more mathematically rigorous and generally yields better results. However, it is also significantly more expensive and infeasible for the large datasets that we use for photometric redshifts. [4] [13] [32] [10]

4.3 Measuring performance

4.3.1 Learning curves

A learning curve tracks the performance of a model with a given recommender. Let $T = \{t_1, \ldots, t_r\} \subset \mathbb{N}$ with $t_1 < \cdots < t_r$ be the counts of training points that we try. Then the learning curve is a function $\ell : T \to \mathbb{R}_{\geq 0}$, whose output is an error.

For a recommender π and a model $F : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \to \hat{\mathcal{F}}$, we generate the learning curve ℓ with algorithm 2. Intuitively, we begin by training F with t_1 random samples and measuring its performance. We then repeatedly add points recommended by π, train the model, and measure its performance. This gives us F's performance as a function of the cardinality of the training.

Algorithm 2: Generating a learning curve.

Data:
A recommender $\pi : \mathcal{P}(\mathcal{X}) \times \mathbb{N} \to \mathcal{P}(\mathcal{X})$,
A function $F : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \to \hat{\mathcal{F}}$ generating a predictor from training samples,
A function $e : \hat{\mathcal{F}} \to \mathbb{R}_{\geq 0}$ yielding the error of a predictor,
A training sample pool $P \subseteq \mathcal{X}$,
Labels for samples in P,
Training example counts to try $T = \{t_1, \ldots, t_r\} \subset \mathbb{N}$ with $t_1 < \cdots < t_r$.

Result: A learning curve $\ell : T \to \mathbb{R}_{\geq 0}$.

$\mathcal{D}_{\mathcal{X}} \leftarrow t_1$ samples drawn randomly from P without replacement
$P \leftarrow P \setminus \mathcal{D}_{\mathcal{X}}$ (remove chosen samples from pool)
$\mathcal{D} \leftarrow \left\{ \left(\mathbf{x}, \tilde{f}(\mathbf{x})\right) \mid \mathbf{x} \in \mathcal{D}_{\mathcal{X}} \right\}$ (label chosen samples)
$\hat{f} \leftarrow F(\mathcal{D})$ (train model)
$\ell(t_1) \leftarrow e\left(\hat{f}\right)$ (compute error)
for $i \in 2, \ldots, r$ **do**
  $\mathcal{D}_{\mathcal{X}} \leftarrow \pi(P, t_i)$ (get recommendations)
  $P \leftarrow P \setminus \mathcal{D}_{\mathcal{X}}$ (remove chosen samples from pool)
  $\mathcal{D} \leftarrow \mathcal{D} \cup \left\{ \left(\mathbf{x}, \tilde{f}(\mathbf{x})\right) \mid \mathbf{x} \in \mathcal{D}_{\mathcal{X}} \right\}$ (label chosen samples)
  $\hat{f} \leftarrow F(\mathcal{D})$ (train model)
  $\ell(t_i) \leftarrow e\left(\hat{f}\right)$ (compute error)
end
return ℓ

We can qualitatively describe the performance of a recommender by examining the learning curve it generates. In general, the better the accuracy of F trained on points given by π, the better π is.

An example of a learning curve can be seen in fig. 4.2. In that figure, the uncertainty recommender clearly outperforms the passive (random) recommender.

4.3.2 Deficiency

The deficiency d is a function $d : \mathfrak{D} \times \mathfrak{F} \times \mathcal{E} \times \mathcal{T} \times \Pi \to \mathbb{R}$. Intuitively, $d(P, F, e, T, \pi)$ measures the room for improvement that a recommender $\pi \in \Pi$ has on a model $F \in \mathfrak{F}$, a training set pool $P \in \mathfrak{D}$, at times $T \in \mathcal{T}$, according to an error function $e \in \mathcal{E}$. In general, the lower the deficiency, the better π performs on our predictor and dataset. [5]

To define $d(P, F, e, T, \pi)$, let $\ell : T \to \mathbb{R}_{\geq 0}$ be the learning curve with a predictor F, a recommender π, a candidate pool P, and an error function e and F at times $T = \{t_1, \ldots, t_r\}$ ordered in ascending order. Generally, ℓ reach its minimum at t_r. This is because, in general, models perform better with more training samples; hence, training on all available samples is likely to yield the best possible model. We use $\ell(t_r)$ as a baseline for a 'perfect' recommender.[(2)]

Then we define d to be the signed area between ℓ and the line $y = \ell(t_r)$, as shown in fig. 4.1. Since ℓ is sampled at discrete intervals, we compute it by applying the trapezoidal rule:

$$d(\pi; P, F, e, T) = \sum_{i=1}^{r-1} \frac{t_{i+1} - t_i}{2} \Big(\big[\ell(t_i) - \ell(t_r)\big] + \big[\ell(t_i) - \ell(t_r)\big] \Big).$$

[(2)] We use the word 'perfect' rather loosely, as it is impossible to define a truly perfect recommender. In most cases, even the best possible recommender will not reach a deficiency of 0 as no proper subset of P will yield comparable performance to the entirety of P.

	Passive	Uncertainty
FVU	0.17	0.16
Mean dz error	0.0202	0.0201
dz bias	−0.00065	−0.0011
dz deviation	0.024	0.023
Fraction of dz outliers	0.0054	0.0046

Table 4.1: Performance breakdown: Passive and uncertainty recommendation from $t = 10000$ to $t = 40000$ with a step size of 100, examined at $t = 12900$. See fig. 4.3 for the plot.

Observe that the deficiency is permitted to be negative. This may occur if F trained on a subset of P with size $t_i < t_r$ performs better than F trained on the entirety of P. Then $\ell(t_i) - \ell(t_r)$ is negative, so the integral is permitted to be negative. This occurs very rarely as most models benefit from larger training sets.

For a fixed, P, F, e, and T, we may compute the deficiencies of recommenders π and π' to compare them. Let ℓ and ℓ' be the learning curves of π and π' respectively. Then $\ell(t_r) = \ell'(t_r)$, since F is always trained on the entire pool P at t_r.[3] Then $d(\pi)$ is the area between ℓ and a baseline, and $d(\pi')$ is the area between ℓ and the same baseline. Since the baseline is the identical, it is possible to directly compare the deficiencies to rank the recommenders.

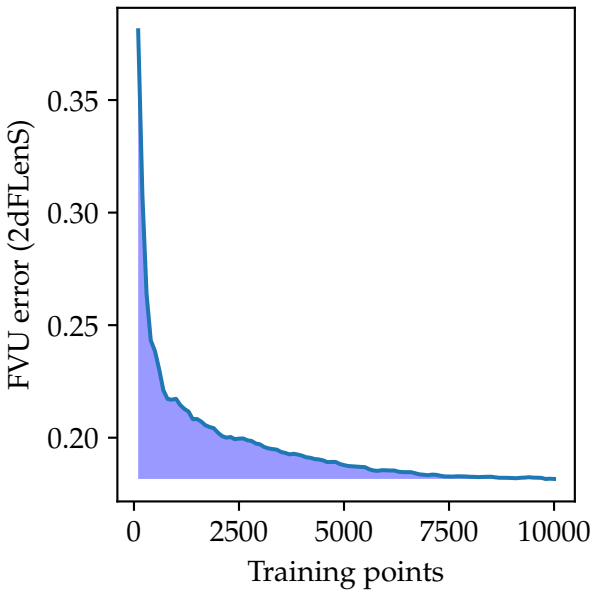


Figure 4.1: A passive learning curve for 2dFLenS. Its deficiency is the area of the shaded region, in this case 135.7.

4.4 Possible recommenders

4.4.1 Passive

The passive recommender is the baseline upon which we wish to improve. It yields random samples from the candidate pool. [2] This makes it equivalent to adding extra training points without utilising any active learning techniques. The passive learning curves we use as a baseline are shown in fig. 3.4, fig. 3.3, and fig. 3.5. We use these to assess the performance of our active learning techniques.

4.4.2 Uncertainty

Let $\operatorname{var} \hat{f} : \mathcal{X} \to \mathbb{R}_{\geq 0}$ be the variance returned by Gaussian process regression. The uncertainty recommender is defined by [18] as a function $\varphi : \mathcal{X} \to \mathbb{R}$ such that

$$\varphi(\mathbf{x}) = \operatorname{var} \hat{f}(\mathbf{x}).$$

Intuitively, it returns the points that we are the most uncertain about. If we are uncertain about a point, then we have less information about that particular area of the feature space.

[3] Whilst this is true mathematically, in reality $\ell(t_r)$ and $\ell'(t_r)$ are approximately but not exactly equal due to numerical inaccuracies. To permit the deficiencies to be compared directly, we find the average of the individual baselines and use it to compute the deficiencies.

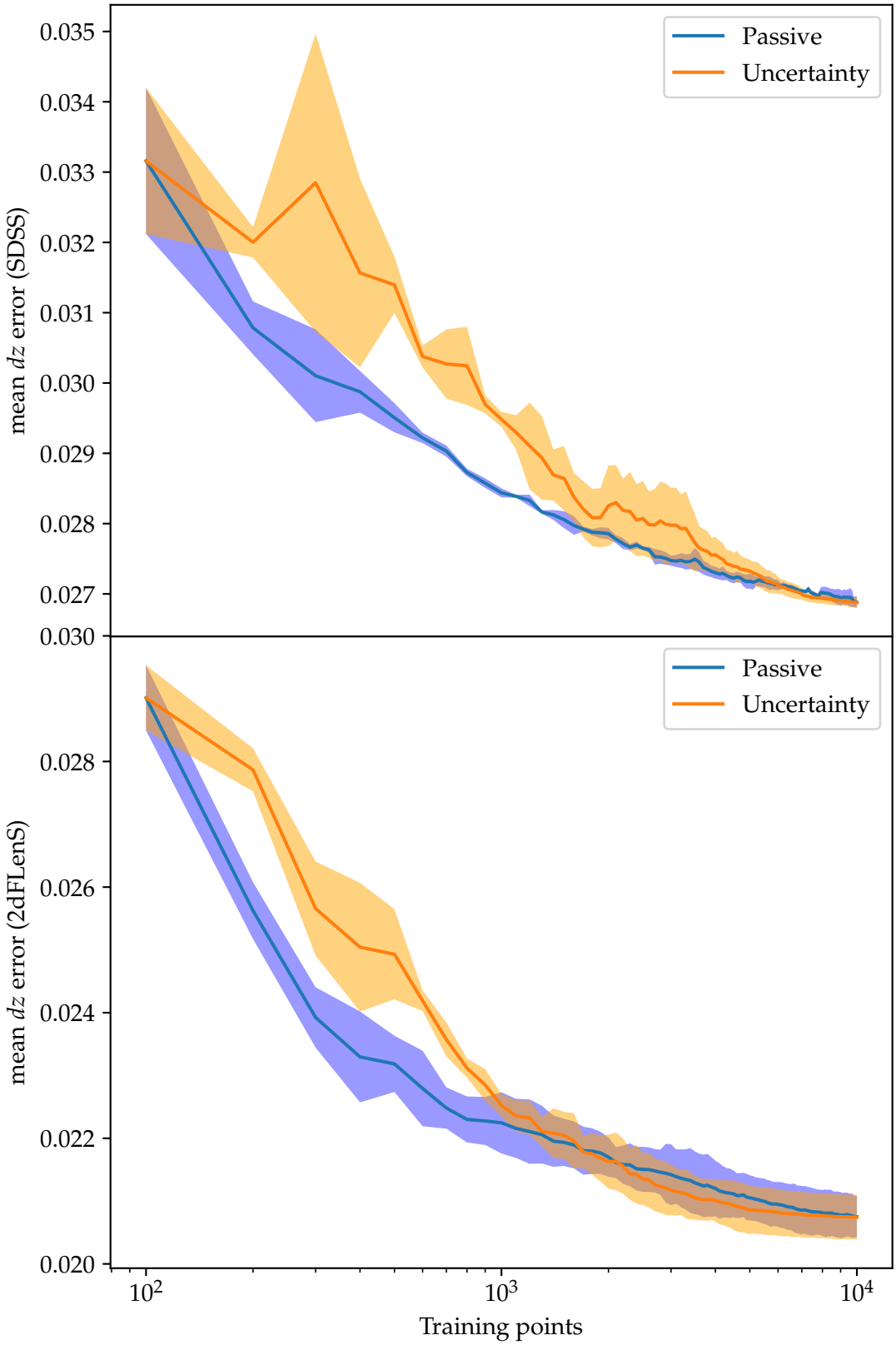


Figure 4.2: Learning curve: mean dz error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen either at random (passive) or using uncertainty recommendation with a step size of 100. The shaded interval represents the one-sigma confidence interval.

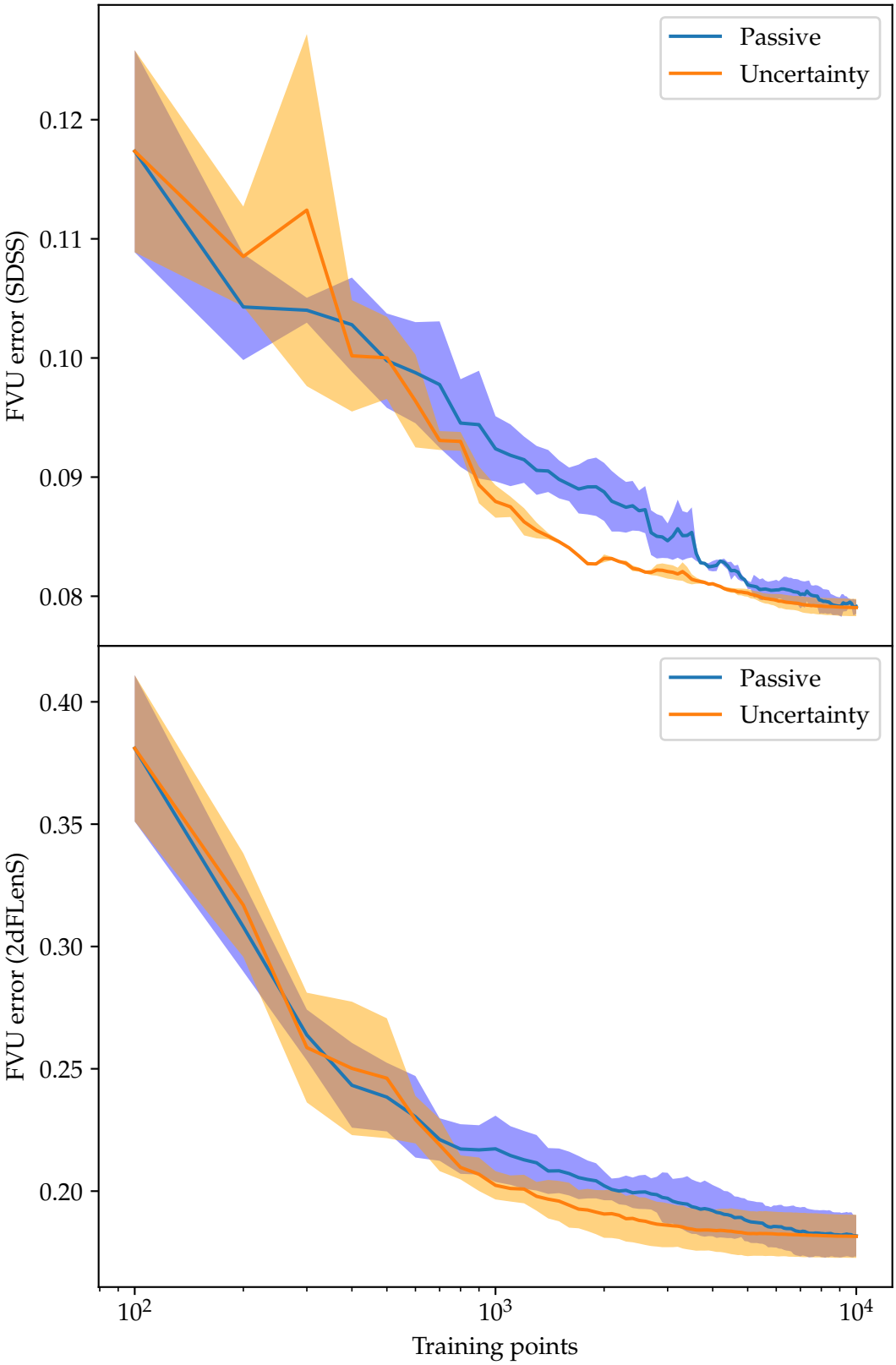


Figure 4.3: Learning curve: FVU error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen either at random (passive) or using uncertainty recommendation with a step size of 100. The shaded interval represents the one-sigma confidence interval.

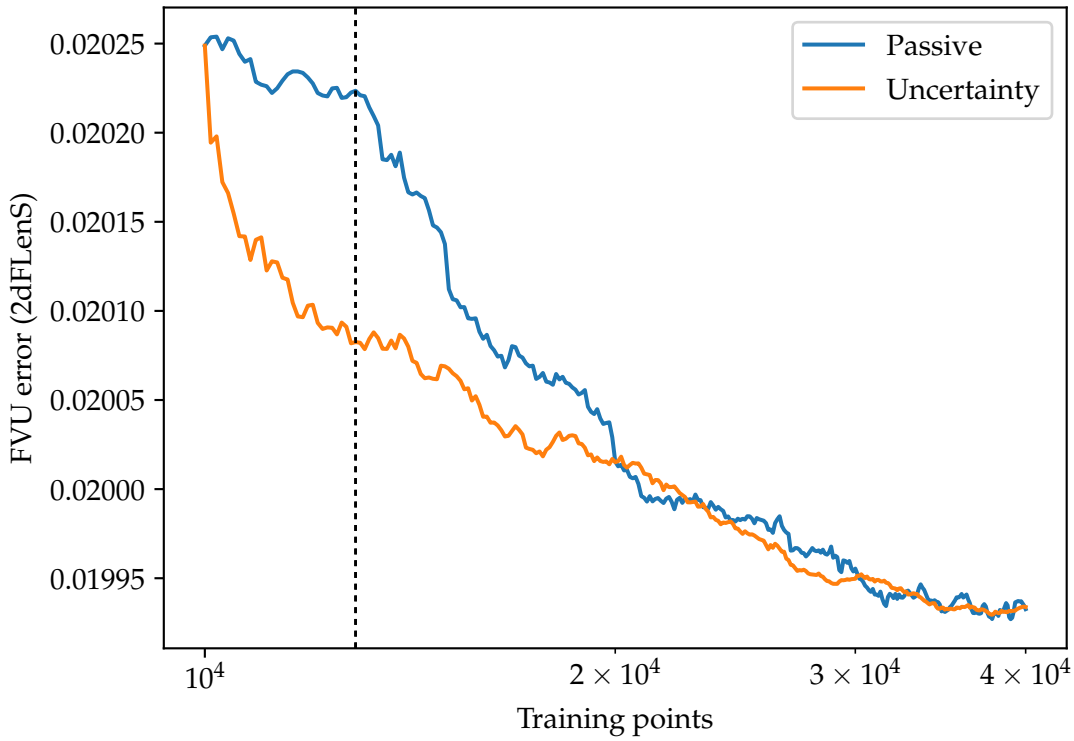


Figure 4.4: Learning curve: FVU error plotted against the number of training points for 2dFLenS. The training points are chosen either at random (passive) or using uncertainty recommendation with a step size of 100. More detailed statistics about the accuracy at $t = 12900$ (marked) see table 4.1.

Hence the points that we are uncertain about would be good additions to the training set.

Learning curves for SDSS and 2dFLenS with the mean dz error are shown in fig. 4.2. Learning curves for the FVU error are shown in fig. 4.3. We see that whereas uncertainty recommendation performs well in the later stages of training, it works rather poorly at the start. This is because we end up focusing on points that we are very unsure about, which tend to be on the long tail of the distribution. Since, in our case, the distribution is five-dimensional, this is a very poor strategy, as we spread out our efforts and neglect the core of the distribution.

We can also approximate this predictor by performing KDE on the labelled set and choosing unlabelled samples lying in the lower-density regions. This is done in [18], achieving lower performance than exact uncertainty recommendation, but better performance than passive recommendation on their dataset.

In fig. 4.4 we see that uncertainty recommendation clearly outperforms passive learning when we begin with a large amount of data. The statistics in table 4.1 show that at when we start with 10 000 points, uncertainty recommendation outperforms passive learning in every metric except bias. This is expected as selectively labelling points can obviously induce selection bias.

4.4.3 Density

Let $D : \mathcal{X} \to \mathbb{R}^+$ be a density function for the dataset. The density estimator is a function $\varpi : \mathcal{X} \to \mathbb{R}$ such that

$$\varpi(\mathbf{x}) := D(x).$$

If we do not know the true distribution of the dataset, we can estimate it. Gaussian mixture models and kernel density estimation are two possible approaches. We use a simple Gaussian with the mean and covariance of our dataset.

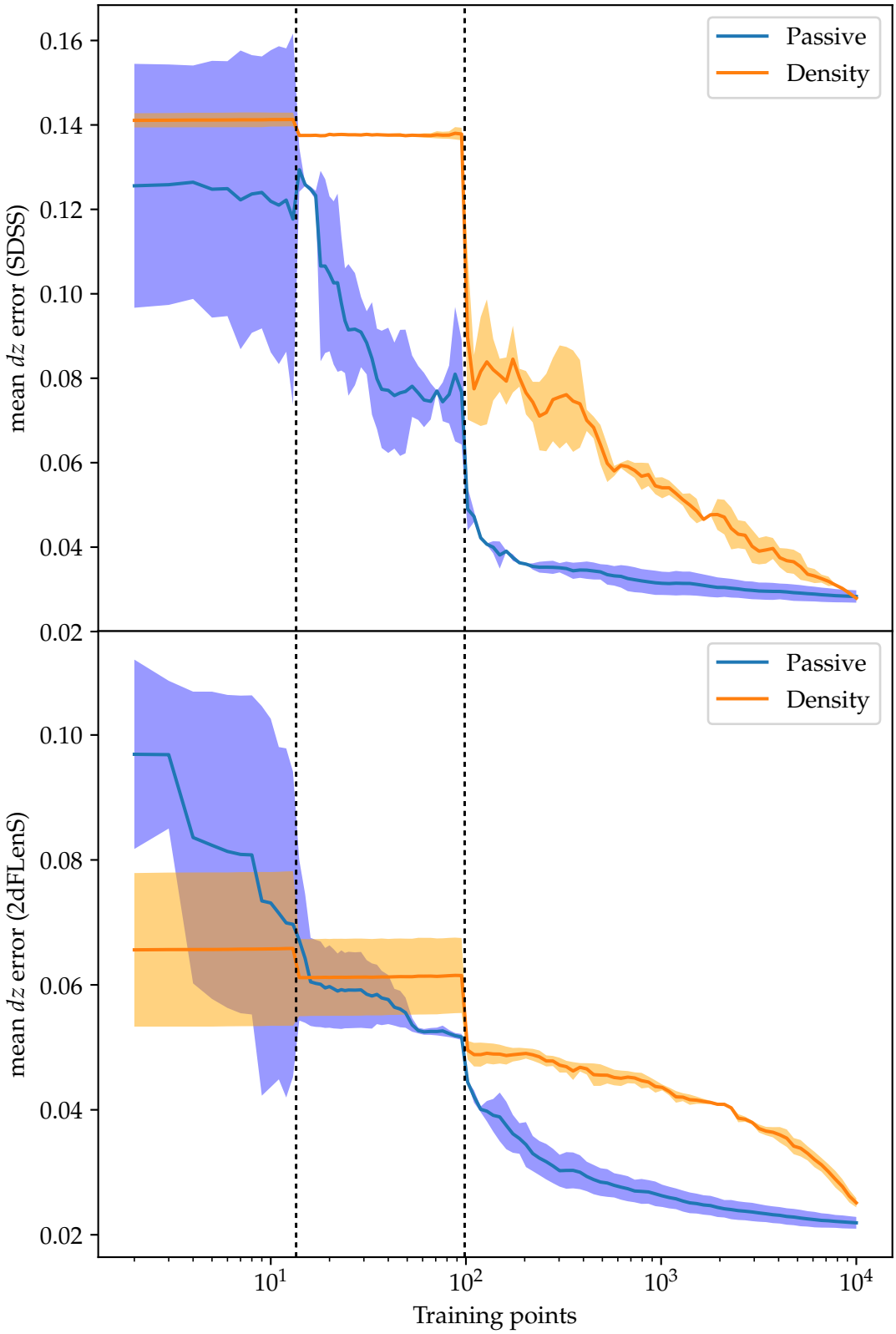


Figure 4.5: Learning curve: mean dz error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen either at random (Passive) or using density recommendation with a logarithmic step size. Each curve begins with two data points chosen by its own recommender. ARD is used to find hyperparameters at $t = 2$, 14, and 100. The density is approximated as a Gaussian. The shaded interval represents the one-sigma confidence interval.

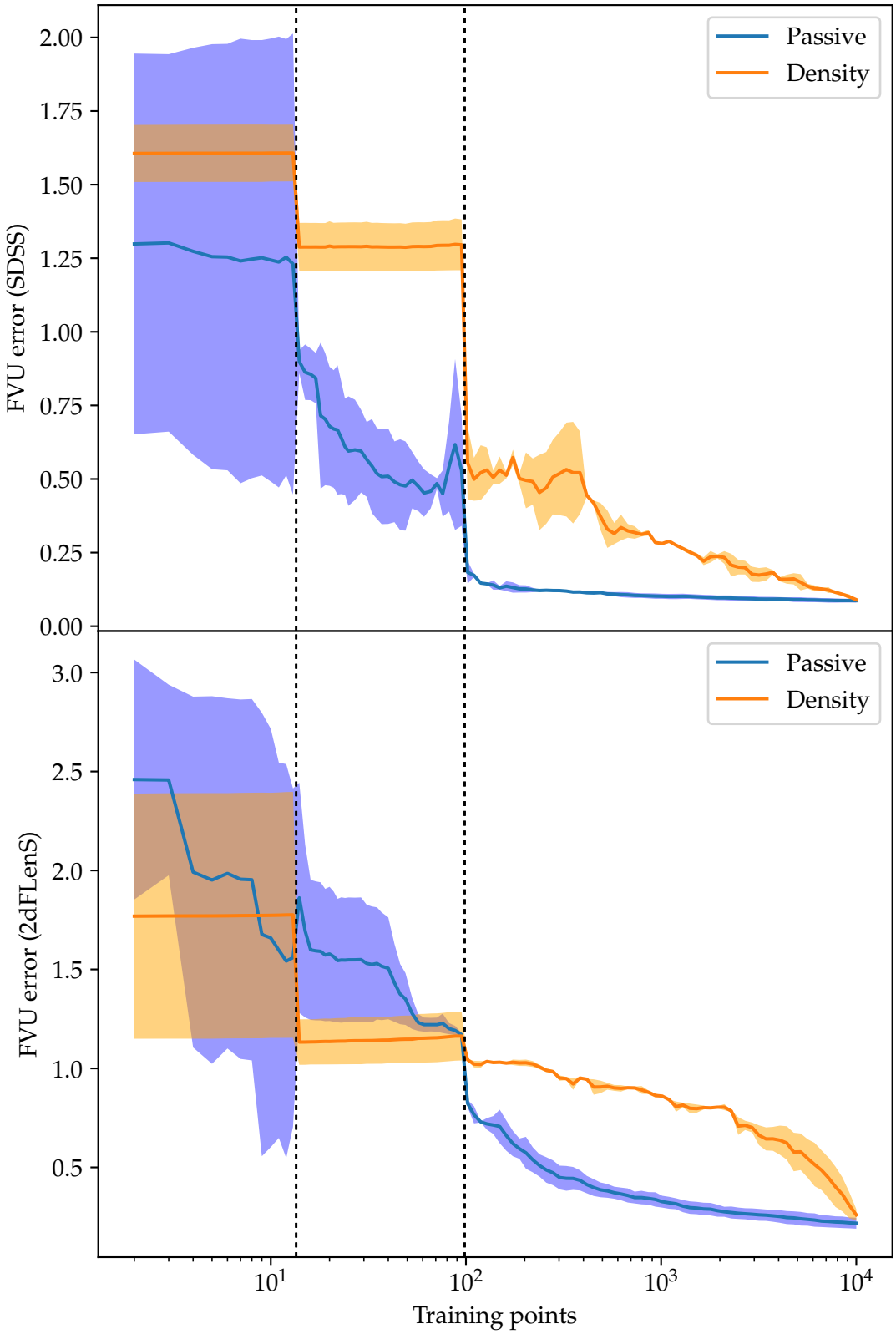


Figure 4.6: Learning curve: FVU error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen either at random (Passive) or using density recommendation with a logarithmic step size. Each curve begins with two data points chosen by its own recommender. ARD is used to find hyperparameters at $t = 2$, 14, and 100. The density is approximated as a Gaussian. The shaded interval represents the one-sigma confidence interval.

	SDSS	2dFLenS
Passive	6.1	5.9
Uncert. (no Boltz.)	8.7	6.2
Uncert. ($T = .01$)	6.4	6.4
Uncert. ($T = .1$)	5.7	4.4
Uncert. ($T = 1$)	6.4	5.6
Uncert. ($T \in [1, .01]$)	5.2	4.9
Density (no Boltz.)	30	39
Density ($T = 1$)	15	25
Density ($T = 10$)	7.1	8.5
Density ($T = 100$)	6.6	6.9
Density ($T \in [10, 100]$)	4.9	7.9

(a) Mean dz error deficiencies

	SDSS	2dFLenS
Passive	48	135
Uncert. (no Boltz.)	31	86
Uncert. ($T = .01$)	26	89
Uncert. ($T = .1$)	24	64
Uncert. ($T = 1$)	39	122
Uncert. ($T \in [1, .01]$)	34	92
Density (no Boltz.)	204	1196
Density ($T = 1$)	124	838
Density ($T = 10$)	47	243
Density ($T = 100$)	36	178
Density ($T \in [10, 100]$)	43	238

(b) FVU error deficiencies

Table 4.2: Deficiency of passive, uncertainty, and density recommenders. Training took place with a step size of 100. Boltzmann sampling is applied to some recommenders, either with a constant temperature or a temperature logarithmically interpolated between the marked values.

We plot the learning curves for SDSS and 2dFLenS under density recommendation and mean dz error in fig. 4.5. Similar curves for the FVU error are shown in fig. 4.6. We note that this recommendation strategy sometimes yields good results at the start of the learning curve (in our case, it works well for 2dFLenS but not SDSS), when we have trained on very few samples. However, it is very quickly overtaken by uncertainty recommendation.

4.4.4 Boltzmann sampling

Consider a physical system that occupies one of M possible states. Let $T > 0$ be the temperature of the system. For each $i = 1, \dots, M$, the state s_i has potential energy ε_i. If $\varepsilon_i < \varepsilon_{i'}$, then we must input energy equalling $\varepsilon_{i'} - \varepsilon_i$ to move from the state s_i to $s_{i'}$. As an example, consider a hydrogen atom consisting of a nucleus and an electron. The electron occupies one of two shells around the nucleus: the inner shell or the outer shell. Energy is required to move this electron the inner shell to the outer shell; hence, the state of lying in the outer shell is of higher energy.

A system seeks to minimise its energy, resulting in a non-uniform probability of states. Higher-energy states are improbable but not impossible, particularly in high temperatures: a sufficiently hot gas will become ionised, entering a higher-energy state. The probability of a state s_i then depends on its energy ε_i and on the temperature T of the system. It follows the *Boltzmann distribution*:

$$p(s_i) = Z \exp\left(\frac{-\varepsilon_i}{k_\mathrm{B} T}\right),$$

where k_B is the Boltzmann constant, and $Z := 1/\sum_{i=1}^{M} \exp(-\varepsilon_i / k_\mathrm{B} T)$ ensures that the probabilities sum to 1. [25]

We can apply these physical principles to active learning to obtain a wider variety of recommendations. Since many scoring functions are continuous, any two recommended samples are likely to be correlated, yielding redundant information. Boltzmann sampling [15] helps reduce this effect.

Let $\varpi : \mathcal{X} \to \mathbb{R}$ be a recommender that assigns a score to the candidate samples. Let $P = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ be the pool of candidates; think of each as a state. Then the energy of $\mathbf{x}_i$ is $\varepsilon_i = -\varpi(\mathbf{x}_i)$ since we prefer higher scores. A Boltzmann sampler with temperature T then

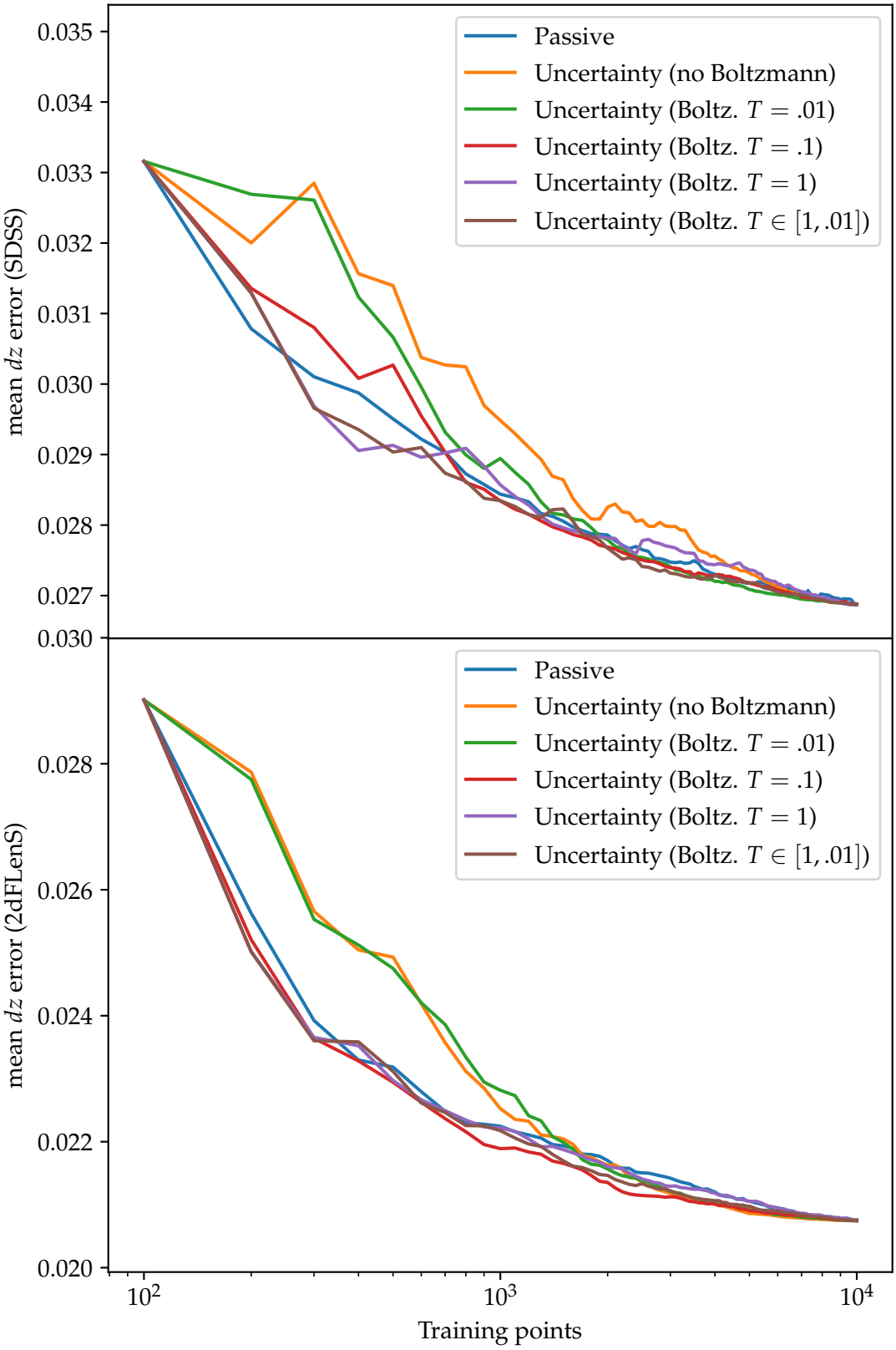


Figure 4.7: Learning curve: mean dz error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen either at random (passive) or using uncertainty recommendation with a step size of 100. Boltzmann sampling is applied to some curves, either with a constant temperature or a temperature logarithmically interpolated between two values.

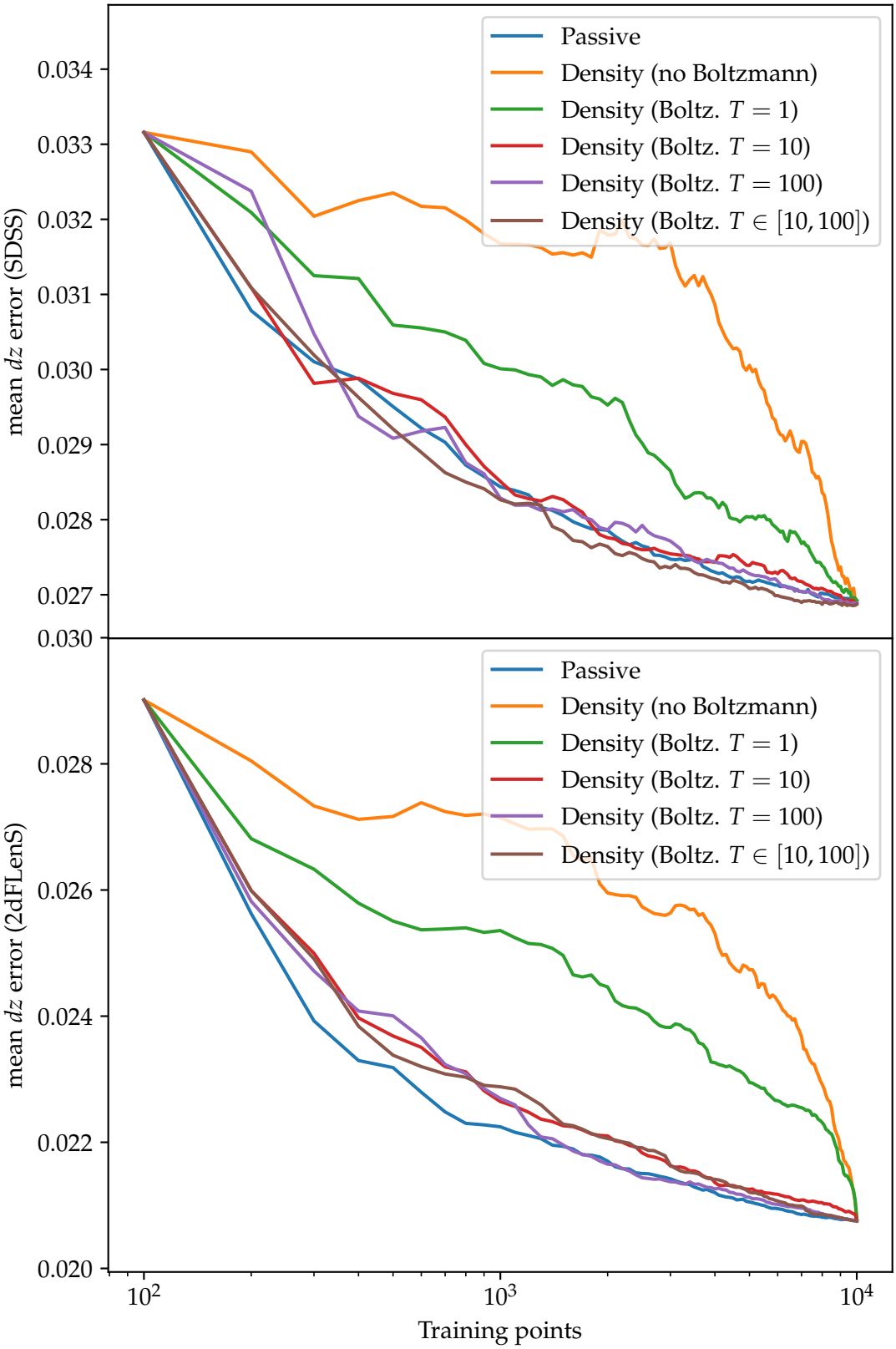


Figure 4.8: Learning curve: mean dz error plotted against the number of training points for both SDSS and 2dFLenS. The training points are chosen either at random (passive) or using density recommendation with a step size of 100. Boltzmann sampling is applied to some curves, either with a constant temperature or a temperature logarithmically interpolated between two values.

recommends a sample $\mathbf{x}_i$ with probability

$$p(\mathbf{x}_i) \propto \exp\left(\frac{-\varepsilon_i}{T}\right) \text{(4)}.$$

Observe that samples with higher scores are still preferred. The strength of that preference decreases as T increases.

We may change the value of T as we progress through our learning to balance exploration and exploitation. Random samples (high T) are often desirable at the start of our learning; towards the end, we can place more weight on φ (low T). Conversely, if a recommender outperforms random samples at the start of our learning curve, we may wish to place more weight on its scores before transitioning to random samples.

A learning curve showing Boltzmann sampling applied to uncertainty recommendation is shown in fig. 4.7. We see that for SDSS, T interpolated from 1 to .01 outperforms passive learning for most of the curve. It also outperforms uncertainty recommendation without Boltzmann. For 2dFLenS, $T = .1$ clearly outperforms both passive learning and pure uncertainty recommendation.

An analogous learning curve for density recommendation is shown in fig. 4.8. For SDSS, we find that T interpolated from 10 to 100 outperforms passive learning. For 2dFLenS, although Boltzmann sampling outperforms pure density recommendation, it does not outperform passive learning.

Table 4.2 lists the deficiencies of these curves. We find that the density recommender with Boltzmann sampling has the lowest mean dz deficiency on SDSS. This is surprising since the density recommender without Boltzmann sampling performs very poorly on SDSS. The same recommender performs significantly worse than passive learning on 2dFLenS.

4.4.5 Mixing recommenders

From the plots above, we see that different recommenders excel at different stages of the training. Density recommendation is best when we are starting off, and uncertainty recommendation helps us tackle the long tail of the distribution in the later stages. The need to combine the benefits of both strategies becomes clear.

There are two ways of combining recommenders, depending on their definition.

If we think of a recommender as a function that assigns a score to a candidate, and we are given two recommenders $\varpi : P \to \mathbb{R}$ and $\varpi' : P \to \mathbb{R}$, then we can derive a third recommender $\tilde{\varpi} : P \to \mathbb{R}$ as

$$\tilde{\varpi}(\mathbf{x}) := kA(\mathbf{x}) + lB(\mathbf{x})$$

for some $k, l \in \mathbb{R}$.

This is not always the most helpful strategy. The distribution of scores returned by ϖ and ϖ' could be totally different, in which case it doesn't make sense to scale each by a constant and sum them. They need to be on the same order and on the same scale.

Instead, another way of deriving a mixed recommender is to think about probabilities. When selecting t elements from $\tilde{\pi}$, we want to select elements from π with probability t and elements from π' with probability $1 - p$. Hence, we can combine pt elements from π and $(1 - p)t$ elements from π' to form the t elements from $\tilde{\pi}$.

A constant p (or equivalently constant k and l) with combine both the advantages and disadvantages of each strategy, yielding a mediocre recommender. We know, however, that uncertainty recommendation and density recommendation excel at different stages of training. It makes sense to prioritise density recommendation at the start, and uncertainty recommendation towards the end. This is reminiscent of the problem of exploration versus exploitation. Uncertainty scores are mostly noise until we have enough data in our training set.

(4) Observe that this formula omits k_B, changing the scale of T.

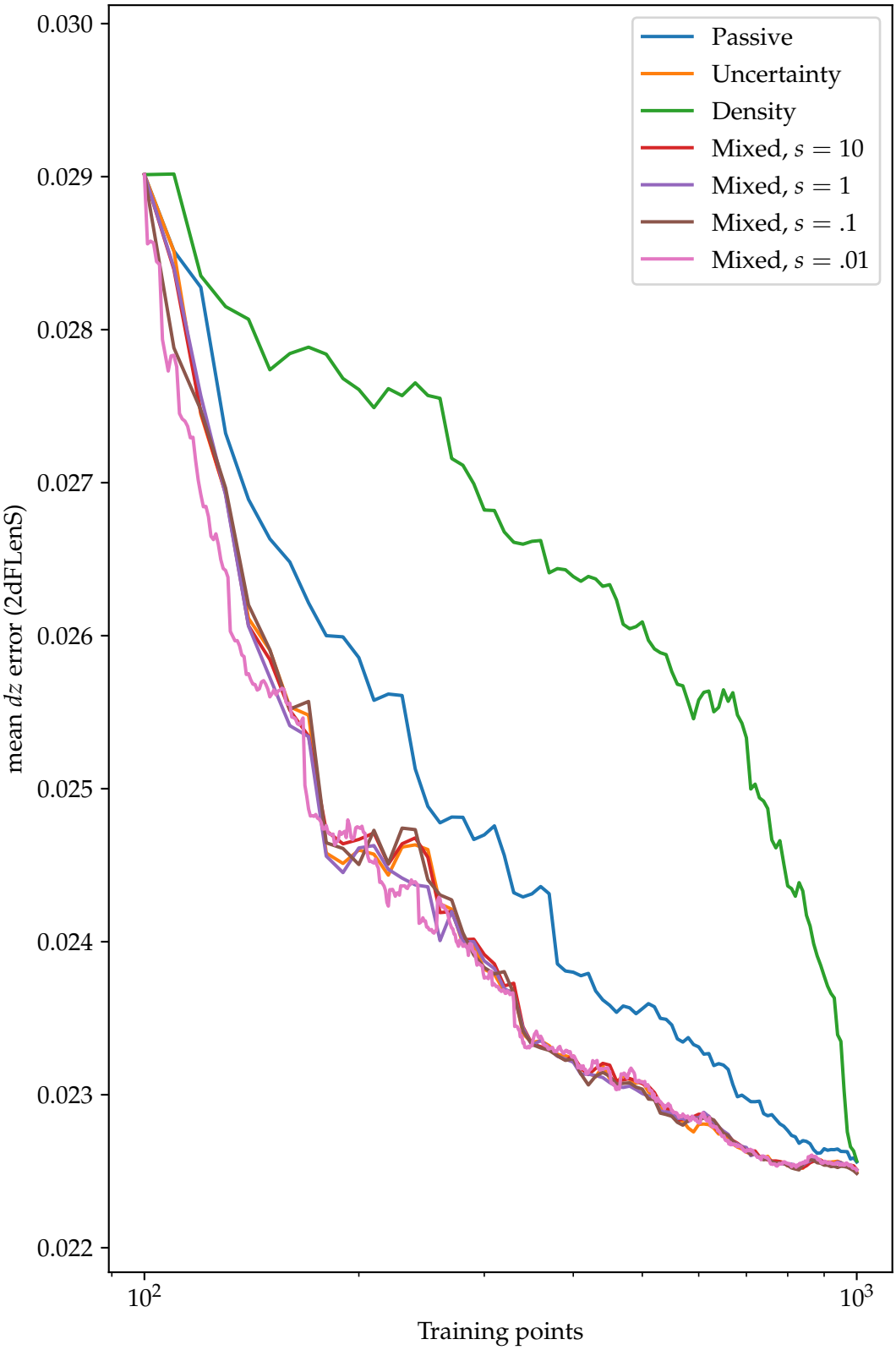


Figure 4.9: Learning curve: mean dz error plotted against the number of training points for 2dFLenS. The training points are chosen either at random (passive), uncertainty recommendation, using density recommendation, or using a mix of uncertainty and density recommendations. Where we use mixed recommendations, st (for t the number of data points in the labelled set) is the ratio of importance of uncertainty recommendation to density recommendation. The step size is 100.

In fig. 4.9 we use the mix

$$\tilde{\varpi} := \varpi + st\varpi',$$

where t is the number of data points labelled so far, ϖ is the density recommender, ϖ' is the uncertainty recommender, and s varies. It can be seen that for lower labelled sets $s = .01$ outperforms uncertainty recommendation. However, this effect is not great and disappears with more data.

Chapter 5

Conclusions

In this thesis, we set out to make it cheaper to improve the accuracy of photometric redshifts using active learning. We discussed the motivation for this endeavour and described our datasets. We developed an online predictor that approximates Gaussian processes with the Gaussian kernel to predict means and confidence intervals, and optimised its hyperparameters. We applied active learning to the problem and found that the uncertainty recommender works very well, since we already have large pools of labelled data.

Further work is still needed. Active learning on regression tasks is a relatively underdeveloped field. A wider range of known recommenders may permit us to improve our performance on this task. It would be useful to have a predictor that does not assume that the cost of labelling every sample is constant; rather, it accounts for it in its recommendations.

Finally, Gaussian process regression is not the perfect tool for photometric redshift prediction, as it does not accept uncertainty bounds on its features. These uncertainty bounds are very common in astronomy may be very useful to the model. Generative models are capable of utilising these, but are expensive. Better regressors for photometric redshifts are hence very desirable.

References

[1] Shadab Alam, Franco D Albareti, Carlos Allende Prieto, Friedrich Anders, Scott F Anderson, Timothy Anderton, Brett H Andrews, Eric Armengaud, Éric Aubourg, Stephen Bailey, et al. The eleventh and twelfth data releases of the Sloan digital sky survey: final data from SDSS-III. *The Astrophysical Journal Supplement Series*, 219(1):12, 2015. 16

[2] Matthew Alger. Learning from crowd labels to find black holes. Honours thesis, The Australian National University, October 2016. 44

[3] Astropy Collaboration, T. P. Robitaille, E. J. Tollerud, P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, A. Ginsburg, A. M. Price-Whelan, W. E. Kerzendorf, A. Conley, N. Crighton, K. Barbary, D. Muna, H. Ferguson, F. Grollier, M. M. Parikh, P. H. Nair, H. M. Unther, C. Deil, J. Woillez, S. Conseil, R. Kramer, J. E. H. Turner, L. Singer, R. Fox, B. A. Weaver, V. Zabalza, Z. I. Edwards, K. Azalee Bostroem, D. J. Burke, A. R. Casey, S. M. Crawford, N. Dencheva, J. Ely, T. Jenness, K. Labrie, P. L. Lim, F. Pierfederici, A. Pontzen, A. Ptak, B. Refsdal, M. Servillat, and O. Streicher. Astropy: A community Python package for astronomy. *Astronomy and Astrophysics*, 558:A33, October 2013. doi: 10.1051/0004-6361/201322068.

[4] Anthony Atkinson, Alexander Donev, and Randall Tobias. *Optimum Experimental Designs, with SAS (Oxford Statistical Science Series)*. Oxford University Press, 2007. ISBN 0199296596. 42

[5] Yoram Baram, Ran El Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5(Mar):255–291, 2004. 43

[6] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *Advances in Neural Information Processing Systems*, pages 1533–1541, 2016. 37

[7] Narciso Benítez. Bayesian photometric redshift estimation. *The Astrophysical Journal*, 536 (2):571, 2000. 24

[8] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, 2013. 35

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2011. ISBN 0387310738. 24

[10] Samprit Chatterjee and Ali S. Hadi. Influential observations, high leverage points, and outliers in linear regression. *Statist. Sci.*, 1(3):379–393, 08 1986. doi: 10.1214/ss/1177013622. URL `https://doi.org/10.1214/ss/1177013622`. 42

[11] Adrian A. Collister and Ofer Lahav. Ann z : Estimating photometric redshifts using artificial neural networks. *Publications of the Astronomical Society of the Pacific*, 116(818):345, 2004. URL `http://stacks.iop.org/1538-3873/116/i=818/a=345`. 24

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics)*. Springer, 2016. ISBN 0387848576. 24

[13] Lior Horesh, Michael Gelbart, Aleksandr Aravkin, and Sergiy Zhuk. Optimal experimental design using gaussian process learning. In *Joint SEG-AGU Summer Research Workshop Advances in Active+Passive "Full Wavefield" Seismic Imaging: From Reservoirs to Plate Tectonics*, January 2014. 42

[14] David JC MacKay et al. Bayesian nonlinear modeling for the prediction competition. *ASHRAE transactions*, 100(2):1053–1062, 1994. 35

[15] Cheng Soon Ong. Acton - a scientific research assistant. URL `https://github.com/chengsoonong/acton`. 50

[16] Cheng Soon Ong. *Kernels: Regularization and optimization*. PhD thesis, The Australian National University, April 2005. 28

[17] Hiroaki Oyaizu, Marcos Lima, Carlos E Cunha, Huan Lin, Joshua Frieman, and Erin S Sheldon. A galaxy photometric redshift catalog for the sloan digital sky survey data release 6. *The Astrophysical Journal*, 674(2):768, 2008. 22

[18] Edoardo Pasolli and Farid Melgani. Gaussian process regression within an active learning scheme. In *2011 IEEE International Geoscience and Remote Sensing Symposium*, 2011. 37, 44, 47

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. , 33, 34, 35

[20] William H. Press. *Numerical Recipes*. Cambridge University Press, 2007. ISBN 0521880688. 37

[21] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009. 33

[22] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2005. ISBN 026218253X. 35

[23] W Romanishin. *An Introduction to Astronomical Photometry Using CCDs*. University of Oklahoma, 2006. 15

[24] Bernhard Schölkopf. Kernel methods. In *Machine Learning Summer School*, Cambridge, 2009. URL `http://mlg.eng.cam.ac.uk/mlss09/mlss_slides/Schoelkopf_1.pdf`. 33

[25] Daniel V. Schroeder. *An Introduction to Thermal Physics*. Pearson, 1999. ISBN 0201380277. 50

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. 42

[27] Donald F Specht. Probabilistic neural networks. *Neural networks*, 3(1):109–118, 1990. 24

[28] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009. 42

[29] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for Gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012. 42

[30] Michalis K Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009. 37

[31] Alasdair Nam Thang Tran. Photometric classification with thompson sampling. Honours thesis, The Australian National University, October 2015. 15

[32] Fabian Triefenbach. Design of experiments: the d-optimal approach and its implementation as a computer algorithm. *Bachelor's Thesis in Information and Communication Technology*, 2008. 42

[33] Chris Williams. Model selection for Gaussian processes. In *Conference on Neural Information Processing Systems*, 2005. 35

[34] C. Wolf, A. S. Johnson, M. Bilicki, C. Blake, A. Amon, T. Erben, K. Glazebrook, C. Heymans, H. Hildebrandt, S. Joudaki, D. Klaes, K. Kuijken, C. Lidman, F. Marin, D. Parkinson, and G. Poole. The 2-degree field lensing survey: photometric redshifts from a large new training sample to r < 19.5. *Monthly Notices of the Royal Astronomical Society*, 466(2):1582–1596, 2017. doi: 10.1093/mnras/stw3151. URL `http://dx.doi.org/10.1093/mnras/stw3151`. 16, 22, 23, 24

[35] Edward L. Wright, Peter R. M. Eisenhardt, Amy K. Mainzer, Michael E. Ressler, Roc M. Cutri, Thomas Jarrett, J. Davy Kirkpatrick, Deborah Padgett, Robert S. McMillan, Michael Skrutskie, S. A. Stanford, Martin Cohen, Russell G. Walker, John C. Mather, David Leisawitz, Thomas N. Gautier III, Ian McLean, Dominic Benford, Carol J. Lonsdale, Andrew Blain, Bryan Mendez, William R. Irace, Valerie Duval, Fengchuan Liu, Don Royer, Ingolf Heinrichsen, Joan Howard, Mark Shannon, Martha Kendall, Amy L. Walsh, Mark Larsen, Joel G. Cardon, Scott Schick, Mark Schwalm, Mohamed Abid, Beth Fabinsky, Larry Naes, and Chao-Wei Tsai. The wide-field infrared survey explorer (wise): Mission description and initial on-orbit performance. *The Astronomical Journal*, 140(6):1868, 2010. URL `http://stacks.iop.org/1538-3881/140/i=6/a=1868`. 16