Identifying fermionic decay signals of the Higgs boson with classification algorithms

Georgina Mansell and Anthony Stephenson

Abstract

Analysing the fermionic Higgs decay channel $H \to \tau\tau$ is important for providing experimental evidence of the Higgs mechanism for fermions. Using statistical classification algorithms we aim to optimise the signal:background ratio in the data (measured by the AMS). In particular, we build a logistic regression model with some engineered features augmenting the original dataset. Using 10-fold cross-validation we select a "best" model using the AMS (scaled by mean absolute deviation over folds) and then test its performance on a hold-out test set. On this set we achieve an AMS of 2.0.

1 Introduction

After the theoretical discovery of the Higgs mechanism and the associated particle, the Higgs boson, experimental confirmation of the prediction became a much sought-after prize in High Energy Physics. Although the theory was more or less accepted by the theoretical community since its inception in the 1960s, experimental evidence was required for verification.

In 2010 CERN's Large Hadron Collider (LHC) achieved the necessary power to probe higher energy regimes of collisions and since then has collected vast amounts of data. In order to verify the theoretical predictions of particle physics, this data needed to be anaylsed carefully to try and distinguish signals from an unknown particle decays from other known decays. The accepted threshold for a discovery in particle physics is the "gold-standard" 5-sigma rule. As a result, experimental discovery of the Higgs was required to meet this threshold.

Various decay mechanisms of the Higgs boson were proposed that could be used to demonstrate its existence. In 2013, when the Higgs was experimentally discovered by CERN, the evidence was provided by *bosonic* decays from the Higgs to the following pairs, $\gamma\gamma$, WW and ZZ. The goal here is to instead examine the coupling of the Higgs to fermions, to verify that their mass can likewise be explained by the Higgs mechanism. Specifically, the aim is to analyse the decay of the Higgs to a pair of tau-flavour leptons $H \to \tau\tau$.

In this document we implement classification models to identify signal events $H \to \tau\tau$ from other background events. An overview of the data set is provided in 2, the classification techniques and performance metrics are introduced in 3, candidate models are compared in 4, and the key findings and validation results are summarised in 5.

2 Data

2.1 Structure

The dataset was provided by CERN for a Machine Learning competition on the website Kaggle. It contains roughly 800,000 proton-proton collision events from the official ATLAS detector simulator. It includes 16 primitive variables measured by the detector, 13 variables derived from the primitive variables, the event classifications, and 4 columns of auxiliary data such as sample weights.

2.2 Class Imbalance

Signal events occur in approximately 1/1000 collisions. If events in the data set were naïvely simulated at their true rate, this would create a highly imbalanced dataset. Classification algorithms can often be modified to account for class imbalance, however as the CERN dataset was simulated the number of samples in each class could be controlled. In the generated dataset approximately 1/3 samples are classified as signal events and the samples are given importance weightings corresponding to their true rate of occurrence. The associated weights of signal and background samples are roughly $O(10^{-3})$ and $O(1)$.

2.3 Missing Data

Early exploration of the dataset revealed the presence of a significant quantity of missing data. Elements of the data matrix recorded as missing were labelled by a value of -999. Before embarking on any model building or even feature engineering we analysed the missing values to try and ascertain whether there was any pattern to their locations. From the definitions of covariates in the dataset it was possible to infer potential causes of missing data. In particular, the estimated mass of the Higgs (DER_mass_MMC) "may be undefined if the topology of the event is too far from the expected topology". Additionally:

- If $PRI\_jet\_num = 1$ the following values are undefined: $DER\_deltaeta\_jet\_jet$, $DER\_mass\_jet\_jet$, $DER\_prodeta\_jet\_jet$, $DER\_lep\_eta\_centrality$, $PRI\_jet\_subleading\_pt$, $PRI\_jet\_subleading\_eta$, $PRI\_jet\_subleading\_phi$
- If $PRI\_jet\_num = 0$ the additional values are undefined: $PRI\_jet\_leading\_pt$, $PRI\_jet\_leading\_eta$, $PRI\_jet\_leading\_phi$

Missing values are therefore not random, and are explained by the combination of an unexpected topology and the number of jets observed. This implies that the physical process is sufficiently different in these regimes that they should be modelled differently in accordance.

3 Method

3.1 Performance Metrics

3.1.1 ROC and AUC

A ROC (Receiver Operating Characteristic) curve plots the true positive rate against the false positive rate for a binary classifier with a varying decision threshold. By doing this we can show the ability of the classifier to dscriminate between the two classes and attain a metric for the performance that we can use to compare models. Chance performance is given by a line at $y = x$ and any curve above this line is an improvement.

The AUC (Area Under the [ROC] Curve) provides a compression of the assessment the ROC curve provides by integrating the ROC curve and giving a value in $[0, 1]$ that represents an overall view of the discriminatory power of the model. A value of 0.5 corresponds to the worst model ($y = x$) whilst 1.0 would represent a perfect model.

For this classification problem, the number of samples of each class do not represent the true ratio of measurements. This means that AUC does not provide a full picture of the model performance, since it only considers the number of true and false positives and does not take into account the samples weights. Instead, the Kaggle competition propose AMS as the objective metric.

3.1.2 AMS

The AMS metric is the Approximate Median discovery Significance; an approximation of the *significance* defined by

$$\begin{aligned} Z &= \Phi^{-1}(1-p) \\ &= \sqrt{q_0} \\ &= \sqrt{2\left(n \ln \frac{n}{\mu_b} - n + \mu_b\right)} \end{aligned}$$

where n is the (unknown) number of events in some search region $\mathcal{G}$, Φ^{-1} is the inverse Normal CDF, q_0 is a test statistic given by Wilks' Theorem and μ_b is the (unknown) expected number of background events. We replace $n \to s + b$ and $\mu_b \to b$ with s, b the estimator of the expected number of signal and background events respectively.

See [**?**] for more details.

Since this is the formal objective we aim to maximise, it might make sense to try and optimise it directly. A little analysis however reveals that the function is non-convex and therefore a poor choice. See A for the full calculation.

To compute the AMS for a given model output $p \in [0, 1]$, we also need to define a *decision rule* that determines the threshold probability at which a signal is declared. For example, if we choose a default threshold, with no prior knowledge at $\theta = 0.5$ then we set our prediction $\hat{y} = s$ if $p > \theta$. In much the same way as we plot the ROC curve by computing the TPR and FPR at different thresholds, so we can plot a curve of AMS against this threshold and thereby determine an optimal value.

3.2 SVM

We considered trying to use a Support Vector Machines as a classification algorithm, and created our own implementation of the algorithm in R. While standard SVM find an optimal linear decision boundary, kernel SVM can project the data into a higher dimensional space and find a more complex boundary. We discuss the Lagrangian optimisation problem for both soft-margin and kernel svm in B.2. Unfortunately SVM was not well suited to this classification problem as the two classes are not well separated. Additionally, this is a large dataset and SVM scales poorly with the number of training samples, and so we are only able to train on a fraction of the data.

Ensemble SVM is a technique where multiple SVM models are trained on subsets of the training data. To classify new samples, the majority vote from the models is taken. Dividing a dataset of n samples over m models reduces the computational complexity of an $\mathcal{O}(n^2)$ algorithm to $\mathcal{O}(n^2/m)$. In Figure 1, we trained 10 soft margin SVM models, each on 500 samples, and classified the remaining samples in each group. We show that the ensemble model outperforms the component models
.

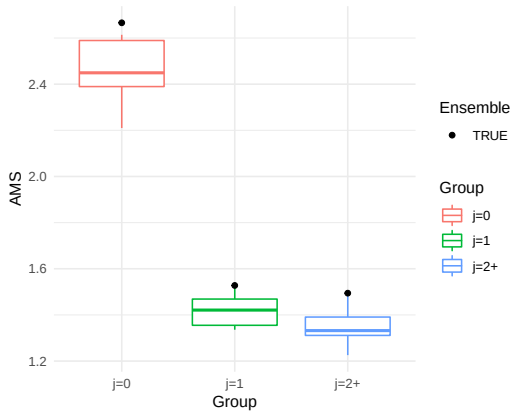


Figure 1: Performance of 10 SVM models and an ensemble SVM per group

3.3 Logistic Regression

We chose to use a logistic regression as a baseline model to try and solve our classification task. The logic for this was two-fold; we would have a robust, flexible model to compare other modelling approaches to, as well as retaining the capacity to augment it with additional features which could include other models as sub-models within the logistic regression model.

3.3.1 Implementation

We implemented our logistic regression model by defining an R class with a standard interface (to enable polymorphism) and an associated fitting function. This function made use of Newton's method (B.1.2) with a backtracking linesearch (B.1.2) in order to calculate maximum likelihood estimates for our model coefficients, $\boldsymbol{\beta}$. B.1 show's

the full calculations of the gradient and Hessian of the logistic likelihood function required to implement Newton's method.

3.4 Feature Engineering

In order to improve the performance of our model, we attempted to carry out some feature engineering to extract as much information as possible from the dataset.

3.4.1 Redundancy

By considering the basic physics of the beam it is possible to see that there is some redundancy amongst the (primitive) features in the dataset. We can exploit this fact and reduce our initial feature space slightly, by transforming the redundant features into a set of new derived features that contain the same information. More specifically, the redundancy comes from the consideration that the physical phenomena should be invariant to certain symmetries; a rotation about the beam (z) axis and reflections in the xy-plane.

In particular, we defined the following new features:

$$
\begin{aligned}
PRI\_lep\_phi - PRI\_tau\_phi &:= (PRI\_lep\_phi - PRI\_tau\_phi) \mod 2\pi \\
PRI\_met\_phi - PRI\_tau\_phi &:= (PRI\_met\_phi - PRI\_tau\_phi) \mod 2\pi \\
PRI\_jet\_leading\_phi - PRI\_tau\_phi &:= (PRI\_jet\_leading\_phi - PRI\_tau\_phi) \mod 2\pi \\
PRI\_jet\_subleading\_phi - PRI\_tau\_phi &:= (PRI\_jet\_subleading\_phi - PRI\_tau\_phi) \mod 2\pi \\
PRI\_tau\_eta &:= \text{sign(PRI\_tau\_eta)PRI\_tau\_eta} \\
PRI\_lep\_eta &:= \text{sign(PRI\_tau\_eta)PRI\_lep\_eta} \\
PRI\_jet\_leading\_eta &:= \text{sign(PRI\_tau\_eta)PRI\_jet\_leading\_eta} \\
PRI\_jet\_subleading\_eta &:= \text{sign(PRI\_tau\_eta)PRI\_jet\_subleading\_eta}
\end{aligned}
$$

3.4.2 Higher-Order Effects

From the exploratory analysis of the data, in particular the visualisation of the principal components, it appears that the classes are not linearly separable in the base feature space. As a result, we hope that perhaps by including non-linear transformations of our features, such as polynomials and RBF centroids, we might be able to model some of the non-linear elements of the relationship between the classes and the features.

RBF Centroids To attempt to capture generic higher order interaction-type effects, we implemented a set of RBF centroid features. The implementation works as follows

Algorithm 1: Augment covariate matrix with RBF centroid features

1. Generate heuristic estimate of RBF hyperparameter s.
2. Select n_c points from $\mathbf{X}_{train}$, $\mathbf{X}_c = \{\mathbf{x}_i^{train}, \ldots, \mathbf{x}_{n_c}^{train}\}$.
3. **for** *i=1:*n_c **do**
   - Calculate $\mathbf{k}_{RBF}(\mathbf{x}_i^{train}, \mathbf{X}_{train}; s)$
   - $\mathbf{X}_{train} \leftarrow \mathbf{X}_{train} \oplus \mathbf{k}_{RBF}(\mathbf{x}_i^{train}, \mathbf{X}_{train}; s)$
4. Fit model.
5. **for** *i=1:*n_c **do**
   - Calculate $\mathbf{k}_{RBF}(\mathbf{x}_i^{train}, \mathbf{X}_{test}; s)$
   - $\mathbf{X}_{test} \leftarrow \mathbf{X}_{test} \oplus \mathbf{k}_{RBF}(\mathbf{x}_i^{train}, \mathbf{X}_{test}; s)$

3.4.3 Polynomial Transformations

In order to try and model non-linear relationships between our features and the target labels, we decided to try and include polynomial transformations of the features. In the results section we will define the (highest) order of the polynomial transformation included in the model by the variable $b \in \{1, 2, 3\}$.

3.4.4 Interactions

Although we considered implementing interactions to further augment our covariates, we chose not to pursue this avenue in the end. The reason for this is that the number of combinations of pairwise interactions for our feature set is 378 Including all of these is likely to significantly increase the risk of overfitting our model, and without a principled method to either choose to only include a subset of interactions, or a way to remove most of them, it seems preferable to skip this option. Additionally, the large increase in feature space would also lead to a sizeable increase in computation time, as the Hessian dimensions would increase by a factor $\sim 4-5$ and hence the time by approximately two orders of magnitude (as our algorithm uses Newton's method which scales as $O(d^3)$ for a d-dimensional feature space) for the case where we include polynomial terms upto third order. If we include only the original features plus interactions, the computation time scales by worse than three orders of magnitude in comparison.

4 Results

4.1 Cross Validation

To make decisions regarding the relative performance of our candidate models we used a 10-fold cross-validation procedure to train and test each of the permutations of our features and models. By running experiments that carried out the procedure, we were able to generate summary outputs that concisely captured a representation of performance. More precisely, we appended rows of information to a *.csv* file for each experiment that could then be subsequently analysed to compare performance. Note that each experiment was run with an unspecified random seed, so we expect some degree of random variation between experiments even with no change to parameters, simply due to the change in data partitioning. We can use this scale

of variance to aid identification of significant performance improvements. 10 fold cross-validation was picked as a compromise between speed of computation and model variance; computation time scales as $O(KNd)$ for K folds, N data points and d features, whilst we expect the variance of our (averaged over folds) parameter estimates to scale as $\frac{1}{K}$. In effect, by picking (for example) 10 folds rather than 5, we are trading an increase in computation time of a factor of two for a decrease in model variance of the same factor, for an equivalent variance in our performance estimate.

4.2 AMS Threshold

Part of our model selection process is to determine the thresholds we want to choose as part of our decision rule to convert probabilistic model outputs into class (signal) predictions. By viewing the curves of AMS against threshold for each fold (2), on each model category, we can see how consistent this relationship is. All three model types show a fairly high amount of variance between folds, although $j = 2+$ appears to be the noisiest. There does appear to be a reasonable degree of agreement between curves however. In order to try and select optimal thresholds, we decided to calculate the minimum over the curves at each threshold, to get an effective lower bounding curve and then find the maximum of this, as a conservative estimate that we hope will minimise the chance of overfitting. A few experiments over different models (varying parameters including n_{rbf} and inclusion of polynomial transformations) we do see that the optimal thresholds are model dependent, which adds to the difficulty. To remove this obstacle, we chose to"integrate out" the threshold as a parameter, by averaging over the results computed over a range of thresholds, in effect, integrating each AMS curve, normalising, and averaging the results.

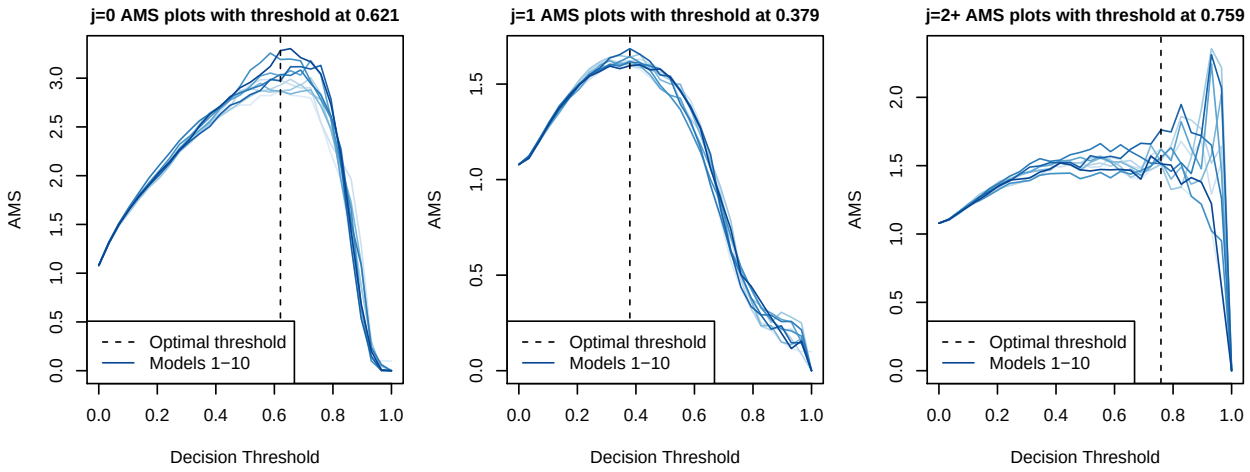


Figure 2: 10-fold cross validation of top model for each jet category

Table 1: Results table of top experiment for each of the groups

G	n_{rbf}	lambda	poly	auc	ams	mad_{ams}	$scaled_{ams}$
1	1	0.00464	3	0.888	2.505	0.097	25.859
2	0	1e-04	3	0.868	1.686	0.053	31.583
3	1	10	3	0.897	2.068	0.122	16.990

4.3 Model Selection

To finally choose our best performing model, we needed to define the criterion with which we want to judge their success. For each experiment, we recorded both the average and mean absolute deviation of the AUC and AMS. After some consideration, we concluded that since our goal was ultimately to maximise AMS, that ought to form the primary benchmark, although experimental results show a high degree of variability between folds. As a result, we opted to scale the AMS measurements by the mean absolute deviation, to try and optimise for the most consistently effective model. The top 5 experiments according to this metric are shown in table 1.

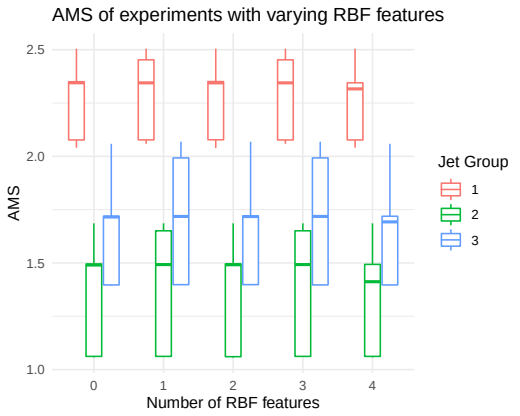
To try and determine how reliable our performance assessment might be, we plotted some figures for both performance measures (AUC and AMS) against varying regularisation, numbers of RBF centroids and order of polynomials. Restricting this summary analysis to our preferred metric, the scaled AMS, we see a noisy but somewhat consistent result in the boxplots in figure 3. The results for n_{rbf} are particularly unclear, with a high degree of variability, whilst the polynomial results, although variable between experiments, show a decisive improvement for 2nd order polynomials. Based on these and the top experiments, we opted for a final model of the following parameters:

$$\begin{aligned} \lambda &= 1 \times 10^{-4} \\ b &= 2 \\ n_{rbf} &= 3 \\ \boldsymbol{\theta} &= (0.6, 0.4, 0.6) \end{aligned}$$

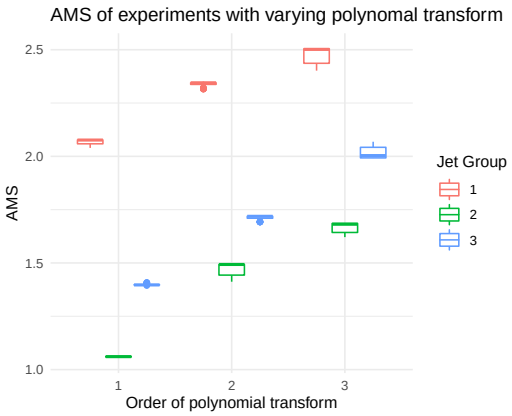
for regularisation parameter λ, polynomial order b, number of RBF centroids n_{rbf} and vector of jet group AMS thresholds $\boldsymbol{\theta} = (\theta_{j=0}, \theta_{j=1}, \theta_{j=2+})$.

4.4 Predictions

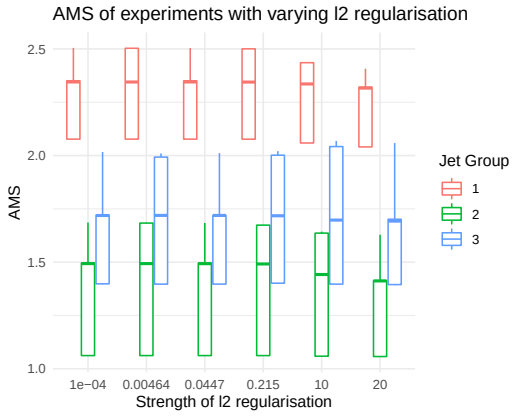
To actually calculate AMS scores, we need to convert out probabilistic outputs into binary labels which means we need to pick a decision rule, i.e. a threshold over which we assign a label 1 (or s) vs 0 (or b). Since we have a different model per jet group, we choose a different optimal threshold per model type and calculate our overall result by computing the average AMS over each group (applying its threshold) and averaging the results. Restating the same thing in more mathematical notation, we



(a) n_{rbf}



(b) Polynomial order



(c) L2 regularisation

Figure 3: Box plots of experimental results varying parameters/features

compute

$$\langle \text{AMS} \rangle = \frac{1}{G} \sum_{g=1}^{G} \frac{1}{K} \sum_{k=1}^{K} \text{ams}(\text{g}, \text{k}, \theta_{\text{g}})$$

with jet groups g, folds k and threshold for group g, θ_g and a function $\text{ams}(\text{g}, \text{k}, \theta_{\text{g}})$ which computes the AMS for a particular group-fold combination given its threshold.

Using the model that ranked the best in terms of scaled AMS, we get the following results on the cross-validated out-of-sample performance for the training set and the hold-out test set are in table 2 where the "error" term here shows the average mean absolute deviation per group across folds.

Clearly the result for the test set is significantly worse than that for the 10-fold out-of-sample measure. A possible explanation can be found by looking at the AMS threshold curves for the two datasets for this model. For the group corresponding to 0 jets, the threshold for the maximum AMS is substantially lower than what we saw in the training set at 0.2 compared to 0.6. The other two jet categories have

Table 2: Results for $\lambda = (0.0046, 0.0001, 10)$, $G = 3$, $n_{rbf} = (1, 0, 1)$, $b = (3, 3, 3)$, K=10 on training set ('t') and validation set ('v')

output	AUC	mad.AUC	AMS	mad.AMS
CV OOS	0.821	0.052	2.029	0.072
Test set	0.833		1.998	

an optimal threshold of 0.5 in the test set, versus 0.4 and 0.6 in the training set.

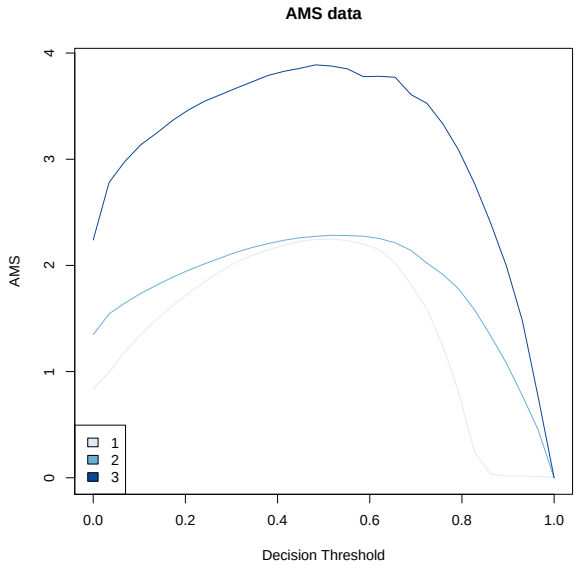


Figure 4: AMS curves for the test set

5 Conclusion

Appendices

A Convexity of the AMS metric

The AMS metric is defined as:

$$\text{AMS} = \sqrt{2\left((s+b)\log(1+\frac{s}{b}) - s\right)}$$

In the documentation, the implication is that in general, we expect $b \gg s$ giving the approximate AMS as $\text{AMS} \sim \frac{\text{s}}{\sqrt{\text{b}}}(1+\mathcal{O}(\frac{\text{s}}{\text{b}}))$.

If we directly check the derivatives of the approximate AMS in the large b regime:

$$\begin{aligned}
\partial_s^2 \frac{s}{\sqrt{b}} &= 0 \\
\partial_b^2 \frac{s}{\sqrt{b}} &= \frac{3}{4}\frac{s}{b^{5/2}} > 0 \\
\partial_{bs}^2 &= -\frac{1}{2b^{3/2}} < 0
\end{aligned}$$

which implies non-convexity (and non-concavity).

To check more carefully though, calculate the terms of the Hessian of the original and then apply the approximation:

$$\begin{aligned}
\partial_s \text{AMS} &= \text{AMS}^{-1}\log\left(1+\frac{\text{s}}{\text{b}}\right) \\
\partial_b \text{AMS} &= \text{AMS}^{-1}\left(\log(1+\frac{\text{s}}{\text{b}}) - \frac{\text{s}}{\text{b}}\right) \\
\partial_s^2 \text{AMS} &= \text{AMS}^{-1}\left(\frac{1}{\text{b}+\text{s}} - \text{AMS}^{-2}\log(1+\frac{\text{s}}{\text{b}})^2\right) \\
\partial_b^2 \text{AMS} &= \text{AMS}^{-1}\left(\frac{\text{s}^2}{\text{b}^2(\text{s}+\text{b})} - \text{AMS}^{-2}(\log(1+\frac{\text{s}}{\text{b}}) - \frac{\text{s}}{\text{b}})^2\right) \\
\partial_{sb}^2 \text{AMS} &= -\text{AMS}^{-1}\left(\frac{\text{s}}{\text{b}(\text{s}+\text{b})} - \text{AMS}^{-2}\log(1+\frac{\text{s}}{\text{b}})(\log(1+\frac{\text{s}}{\text{b}}) - \frac{\text{s}}{\text{b}})\right)
\end{aligned}$$

and then verify whether any of them are ever < 0.

Using the AMS approximation, and $\log\left(1+\frac{s}{b}\right) \sim \frac{s}{b} - \frac{1}{2}\left(\frac{s}{b}\right)^2$ we get the following

$$\begin{aligned}
\partial_s^2 \text{AMS} &\sim \frac{\sqrt{b}}{s}\left[\frac{1}{(b+s)} - \frac{b}{s^2}(\frac{s^2}{b^2} - \frac{s^3}{b^3})\right] \\
&= \frac{\sqrt{b}}{s(s+b)} - \frac{1}{s\sqrt{b}} + \frac{1}{b^{3/2}} \\
&\gtrsim \epsilon
\end{aligned}$$

Since in this regime of large b this term is negative, the Hessian cannot be positive definite and hence the metric is non-convex. If the Hessian is *concave* though, we can simply optimize $-$AMS, so we need to check the other terms.

$$\begin{aligned}\partial_b^2 \text{AMS} &\sim \frac{\sqrt{b}}{s}\left[\frac{s^2}{b^2(s+b)} - \frac{b}{s^2}(\frac{1}{4}\frac{s^4}{b^4})\right] \\ &= \frac{s}{b^{3/2}(s+b)} - \frac{1}{4}\frac{s}{b^{5/2}} \\ &\gtrsim \epsilon\end{aligned}$$

$$\begin{aligned}\partial_{bs}^2 \text{AMS} &\sim -\frac{\sqrt{b}}{s}\left[\frac{s}{b(s+b)} + \frac{b}{s^2}(\frac{s}{b} - \frac{1}{2}\frac{s}{b^2})(-\frac{1}{2}\frac{s^2}{b^2})\right] \\ &= \frac{1}{2b^{3/2}}(1 - \frac{1}{2}s) - \frac{1}{\sqrt{b}(s+b)} \\ &< 0\end{aligned}$$

where we take ϵ to be some suitably small positive number.

If we decide to take some tolerance, $\epsilon \sim 10^{-6}$ at which to ignore higher order terms in $\frac{s}{b}$ we can check specifically that the claims on the Hessian above hold in that regime.

So, let's assume we can ignore terms of $O(\frac{s}{b})^3$ then for $\epsilon \sim 10^{-6}$ a ratio of $\frac{s}{b} \sim 10^{-2}$ would suffice; so to test this, let us choose $s = 1$ and $b = 100$ for simplicity. With this, we find that $\partial_s^2 \text{AMS} \sim +10^{-6}$, $\partial_b^2 \text{AMS} \sim +10^{-6}$ and $\partial_{bs}^2 \text{AMS} \sim -10^{-4}$ which satisfies the claims above. Finally, to verify that this is a reasonable assertion, if we check the ratio of $s : b$ in the data (using the weight vector) we find a ratio of approximately 10^{-3}, which means our assumption was conservative.

B Modelling

B.1 Newton's Method for Logistic Regression

From the conditional distribution under the logistic regression model for a single point $\mathbf{x} \in R^d$, $y \in \{+1, -1\}$ and coefficients $\boldsymbol{\beta} \in R^d$.

$$\begin{aligned}p(y \mid \mathbf{x}, \boldsymbol{\beta}) &= \sigma(f(\mathbf{x}; \boldsymbol{\beta}) \cdot y) \\ &= (1 + \exp(y\langle \mathbf{x}, \boldsymbol{\beta}\rangle))^{-1}\end{aligned}$$

We want to minimise the negative loglikelihood:

$$\begin{aligned}l(\mathbf{x}, y, \boldsymbol{\beta}) &= \log(1 + \mathrm{e}^{y\langle \mathbf{x}, \mathbf{x}\rangle}) \\ \partial_{\beta_a} l &= \frac{y x_a \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle}}{1 + \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle}} \\ \partial_{\beta_a \beta_b}^2 l &= \frac{y^2 x_a x_b \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle}}{(1 + \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle})^2}\left(1 + \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle} - \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle}\right) \\ &= \frac{x_a x_b \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle}}{(1 + \mathrm{e}^{y\langle \mathbf{x}, \boldsymbol{\beta}\rangle})^2} \\ &= x_a x_b w(x)\end{aligned}$$

where in the last line we define a weight function $w(x) = \frac{\mathrm{e}^{-x}}{(1+\mathrm{e}^{-x})^2}$ (and the elementwise vector version, $\mathbf{w}(\mathbf{x}) = \frac{\mathrm{e}^{-\mathbf{x}}}{(1+\mathrm{e}^{-\mathbf{x}})^2}$).

To implement Newton's method we want to try and simplify the expressions. First define the logistic and logit functions as

$$\gamma(x) = (1 + \mathrm{e}^{-x})^{-1}$$
$$\gamma^{-1}(p) = \log\left(\frac{p}{1-p}\right)$$

and then write out the derivatives in terms of the logistic function:

$$\partial_{\beta_a} l(y = +1) = x_a \gamma(\langle \mathbf{x}, \boldsymbol{\beta} \rangle)$$
$$\partial_{\beta_a} l(y = -1) = -x_a (1 - \gamma(\langle \mathbf{x}, \boldsymbol{\beta} \rangle)$$

If we then map $y \in \{1, -1\}$ to $y' \in \{0, 1\}$ we can condense these into a single expression

$$\partial_{\beta_a} l(y) = -x_a (y - \gamma(\langle \mathbf{x}, \boldsymbol{\beta} \rangle))$$

and rewrite the loss as $l(y') = \log(1 + \mathrm{e}^{(-1)^{y'} \langle \mathbf{x}, \boldsymbol{\beta} \rangle})$.

Now for our entire dataset D, assumed to be IID, we have a total loss, gradient and Hessian

$$\begin{aligned} L &= \sum_{i \in D} l_i = \sum_{i \in D} \log(1 + \mathrm{e}^{y_i \langle \mathbf{x}_i, \boldsymbol{\beta} \rangle}) \\ \partial_{\boldsymbol{\beta}} L &= -\sum_{i \in D} \mathbf{x}_i (y_i - \gamma(\langle \mathbf{x}_i, \boldsymbol{\beta} \rangle)) \\ &= -\mathbf{X}^T (\mathbf{y} - \gamma(\mathbf{X}\boldsymbol{\beta})) \\ \boldsymbol{H} &= \sum_{i \in D} \mathbf{x}_i \mathbf{x}_i^T w(\mathbf{x}_i) \\ &= \mathbf{X}^T \boldsymbol{W} \mathbf{X} \end{aligned}$$

with $\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{pmatrix} \in R^{n \times d}$, $\boldsymbol{W} = \mathrm{diag}(\mathbf{w}(\mathbf{x})) \in R^{n \times n}$.

B.1.1 Regularisation

We can add constraints to our minimisation problem, to help constrain the coefficients and effectively implement L_2 regularisation by formulating the following convex optimisation problem.

$$\begin{aligned} \min. \quad & L \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|^2 \leq c \end{aligned}$$

From this we can construct the Lagrangian $\mathcal{L} = L + \lambda(\|\boldsymbol{\beta}\|^2 - c)$ and hence augment the gradient and Hessian we calculated above as

$$\partial_{\boldsymbol{\beta}} \mathcal{L} = \partial_{\boldsymbol{\beta}} L + 2\lambda\boldsymbol{\beta}$$
$$\tilde{\boldsymbol{H}} = \boldsymbol{H} + 2\lambda\mathbf{I}$$

Since we have a free parameter c that determines the optimal value $\lambda^\star(c)$ of our dual variable, we can equivalently ignore c and treat λ as a parameter to be tuned.

B.1.2 Newton's method

Algorithm 2: Newton's method

1. Initialise parameters; $\boldsymbol{\beta}_0$, λ_0, ϵ, $L_0 = L(\boldsymbol{\beta}_0)$, m
2. **for** *i=1:m* **do**
   - Compute $\partial_{\boldsymbol{\beta}} L$ and $\boldsymbol{H}$
   - Calculate Newton step $\Delta\boldsymbol{\beta} = -\boldsymbol{H}^{-1}\partial_{\boldsymbol{\beta}} L$
   - Calculate stepsize α with backtracking linesearch (B.1.2)
   - Update $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} + \alpha\Delta\boldsymbol{\beta}$
   - Compute new loss L
   - Test stopping criterion: $|L - L_0| < \epsilon$
3. Return $\hat{\boldsymbol{\beta}}$.

Algorithm 3: Backtracking linesearch

1. Initialise parameters; γ, τ
2. Set largest stepsize $s_{max} \leftarrow 1$
3. Set $s \leftarrow s_{max}$
4. **while** $f(x + \Delta x) > f(x) + \gamma s \nabla f^T \Delta x$ **do**
   - Update $s \leftarrow \tau s$
5. Return s

B.2 SVM

B.2.1 Soft-margin SVM

For a dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x} \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$, support vector machines (SVM) aim to find the optimal hyperplane which separates samples of each class y. The optimal hyperplane $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x} + w_0 = 0$ classifies all samples correctly and has the largest distance to the nearest sample. The parameters $\mathbf{w}$ and w_0 are found by solving the following optimisation problem:

$$\begin{aligned} \text{minimise: } & \frac{1}{2}||\mathbf{w}||^2 \\ \text{subject to: } \forall_i, \quad & y_i f(\mathbf{x}_i, \mathbf{w}) \geq 0 \end{aligned}$$

When classes are not linearly separable, soft-margin SVM allows some samples to be misclassified by the boundary by including slack variables ϵ_i. The slack variables are penalised, and parameter C controls the strength of this penalty compared to the margin. The optimisation problem becomes:

$$\begin{aligned} \text{minimise: } & \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \epsilon_i \\ \text{subject to: } \forall_i, \quad & y_i f(\mathbf{x}_i, \mathbf{w}) + \epsilon_i \geq 1, \quad \epsilon_i \geq 1 \end{aligned}$$

The problem can be re-written as a Lagrangian:

$$L(\boldsymbol{\lambda}, \boldsymbol{\eta}) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \epsilon_i - \sum_i \lambda_i(y_i(\mathbf{w}^T\mathbf{x} + w_0) + \epsilon_i - 1) - \sum_i \eta_i\epsilon_i$$

Setting the differentials to 0 gives the following conditions:

$$\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i^T, \qquad \sum_i \lambda_i y_i = 0, \qquad \lambda_i + \eta_i = C$$

and substituting these back into the Lagrangian eliminates $\mathbf{w}, w_0, \epsilon_i$ and η_i to give:

$$L(\boldsymbol{\lambda}) = -\frac{||\sum_i \lambda_i y_i \mathbf{x}_i^T||^2}{2} + \sum_i \lambda_i$$

This leaves a quadratic optimisation problem to find $\boldsymbol{\lambda}$ subject to the KKT conditions:

$$0 \leq \lambda_i \leq C, \qquad \sum_i \lambda_i y_i = 0$$

To use a quadratic optimisation package, it is useful to rewrite the problem in the form $\boldsymbol{\lambda}^T \mathbf{D} \boldsymbol{\lambda}$. Letting $\mathbf{X}$ be a matrix with rows $\mathbf{x}_i^T$, $\mathbf{y}$ be an $n \times 1$ vector, and $\circ$ be a Hadamard product:

$$L(\boldsymbol{\lambda}) = -\frac{\boldsymbol{\lambda}((\mathbf{y}\mathbf{y}^T) \circ (\mathbf{X}\mathbf{X}^T))\boldsymbol{\lambda}}{2} + \langle \mathbf{1}, \boldsymbol{\lambda} \rangle \tag{1}$$

Once $\boldsymbol{\lambda}$ is found, $\mathbf{w}$ can be retrieved using $\mathbf{w} = \sum_i \lambda_i y_i \mathbf{x}_i^T$, and w_0 can be retrieved using the support vectors. Let $\mathcal{S}$ be the set of support vectors: samples where $\lambda_i \neq 0$ and $y_i(\mathbf{w}^T \mathbf{x} + w_0) = 1$. Any support vector can be used to find w_0, but it is more numerically stable to find the average across the set:

$$w_0 = \frac{1}{|\mathcal{S}|} \sum_{m \in \mathcal{S}} y_m - \mathbf{w}^T \mathbf{x}_m$$

Given a new sample $\mathbf{x}_n$, it's class is predicted as:

$$\hat{y}_n = \text{sign}(\mathbf{w}^T \mathbf{x}_n + w_0)$$

B.2.2 Kernel SVM

In equation 1 the input data appears only in the term $\mathbf{X}\mathbf{X}^T$, which is equivalent to a linear kernel matrix $\mathbf{K}$, where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$. Other kernel functions can also be used such as a b-degree polynomial kernel $(\mathbf{x}_i^T \mathbf{x}_j + 1)^b$.

Given a new sample $\mathbf{x}_n$, it's class is predicted as:

$$\hat{y}_n = \text{sign}\Big(\sum_{m \in \mathcal{S}} \lambda_m y_m k(\mathbf{x}_n, \mathbf{x}_m) + w_0 \Big)$$

where w_0 is calculated as:

$$w_0 = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \Big(y_s - \sum_{m \in \mathcal{S}} \lambda_m y_m k(\mathbf{x}_s, \mathbf{x}_m) \Big)$$