ST4299 Report

Intern: Wong Tau Yew, Mark (A0190753B)

Table of Contents

1. Introduction

Objective

The aim of this report is to replicate certain results and strategies presented in "Empirical Asset Pricing via Machine Learning by Shihao Gu, Bryan Kelly, Dacheng Xiu" as well as to extend the

analysis to other models. The authors of the paper used numerical models to predict the monthly returns of stocks with data from 1957 to 2020. These models include ordinary least squares (OLS), elastic net, partial least squares (PLS), principal component regression (PCR), generalised linear model (GLM), random forest (RF), gradient-boosted random trees (GBRT) and neural networks (NN) with 1-5 hidden layers.

Data preparation

Similar to the paper, data is considered on a monthly basis partly due to cases of lossy data for certain stocks which made it hard for further dividing into weekly or daily. Due to limited computational resources, 84 out of 94 stock characteristics detailed in the paper are included. Original data required for computation of stock characteristics are obtained separately from The Center for Research of Security Prices (CRSP) and Compustat and Institutional Brokers' Estimate System (IBES). Code to fetch various tables through Wharton Research Data Services (WRDS) and calculate stock characteristics is adapted from Jeremiah Green's SAS code in his website[1]. 8 macroeconomic predictors are obtained directly or via simple calculations from data supplied through Amit Goyal's website[2]. To reduce computational expense, Standard Industrial Codes (SIC) are dropped instead of hot-encoded into dummy variables. This totals for 84+8=92 predictors. Names and abbreviations of all predictors used can be found in Appendix C. For more details on stock characteristics, refer to the supplementary data of the paper.

The final data range is constrained from 1994 to 2020. Missing values in the data are filled with the cross-sectional (observations of the same time period) median of all stocks. This is preferable to forward-filling observations by stock since the characteristics are generally highly variable. An alternate experimentation with the difference of forward-filling observations by stock and filling the rest (e.g. missing values for momentum when a stock is initially listed) with 0 is done and results generally have shown to be comparatively worse. Stock characteristics which involve percentage changes may contain infinite values stemming from division by 0 especially from the onset of certain variables. The positive infinite and negative infinite values are truncated to the maximum and the minimum values of the characteristic respectively. Altogether, the total number of observations is 1429680. The response variable (y) is designated as the realised returns (ret) for the next month specific to each stock. This reduces the number of observations to 1076826.

Equity returns are calculated based on prices adjusted for corporate actions. Specifically, the prices are adjusted according to stock splits and dividend distribution to better reflect the value to an investor. For example, if a company decided on a 2:1 stock split, the stock price is reduced by 50% while a current investor will be issued new shares and holding on to twice the number of stocks as before. In entirety, the total value of the investment remains the same for the investor. In the case of dividend distribution, when a dividend is declared, the stock price is driven up prior to dividend distribution, usually by about the same amount as dividend received per unit

[1] Jeremiah Green's website is 'https://sites.google.com/site/jeremiahrgreenacctg/home'. Annual Consumer Price Index (CPI) required for certain computations can be found in 'https://www.bls.gov/cpi/tables/supplemental-files/home.htm'. Adapted python code is attached.
[2] Amit Goyal's website is 'https://sites.google.com/view/agoyal145'.

share. This change in stock price is reverted after the distribution. During the period leading up to the distribution from the point of declaration, the stock price is adjusted for inflation.

2. Exploratory data analysis

Let p denote adjusted closing prices. For month t,

$$\mathrm{p}_{stock,t} = \frac{|\mathrm{prc}_{stock,t}|}{\mathrm{cfacpr}_{stock,t}}$$

In the case when no trade (vol=0) is performed for a particular day, the stock price is represented virtually as the average of the bid and ask price. This stock price is recorded as negative in prc to differentiate from stock prices that are actively traded. For this reason, the absolute value of prc is used in the formula above.

$$\begin{aligned} returns_{stock,t} &= \frac{\frac{|\mathrm{prc}_{stock,t}| + \mathrm{divamt}_{stock,t}}{\mathrm{cfacpr}_{stock,t}}}{p_{stock,t-1}} - 1 \\ &= \frac{p_{stock,t} + \frac{\mathrm{divamt}_{stock,t}}{\mathrm{cfacpr}_{stock,t}}}{p_{stock,t-1}} - 1 \end{aligned}$$

Monthly returns are similarly precomputed and obtained directly from the database.

Definitions for some stock characteristics highly ranked for importance in the paper are provided below.

Momentum

$$\mathrm{mom1m}_{stock,t} = 1 + returns_{stock,t-1} - 1 = returns_{stock,t-1}$$

$$\mathrm{mom6m}_{stock,t} = \prod_{i=2}^{6}(1 + returns_{stock,t-i}) - 1$$

$$\mathrm{mom12m}_{stock,t} = \prod_{i=2}^{12}(1 + returns_{stock,t-i}) - 1$$

$$\mathrm{mom36m}_{stock,t} = \prod_{i=13}^{36}(1 + returns_{stock,t-i}) - 1$$

Change in 6-months momentum (chmom)

$$\begin{aligned} \mathrm{chmom}_{stock,t} &= \prod_{i=1}^{6}(1 + returns_{stock,t-i}) - 1 - (\prod_{i=7}^{12}(1 + returns_{stock,t-i}) - 1) \\ &= \prod_{i=1}^{6}(1 + returns_{stock,t-i}) - \prod_{i=7}^{12}(1 + returns_{stock,t-i}) \end{aligned}$$

Size/Market capitalization (mvel1)

$$\mathrm{mvel1}_{stock,t} = log(p_{stock,t-1} * \mathrm{shrout}_{stock,t-1})$$

Maximum daily returns (maxret)

$$\text{maxret}_{stock,t} = max(returns_{stock,t,1}, returns_{stock,t,2}, \ldots, returns_{stock,t,\text{maximum day in month}})$$

Industrial momentum (indmom)

$$indmom_{industry,t} = \frac{\sum_{i=1}^{n} I_{stock \in industry} mom12m_{stock,t}}{n_{industry}}$$

The first 2 digits of siccd (sic2) are used for the classification of stocks according to industry.

Volatility of returns (retvol)

$$retvol = var(returns_{stock,t,1}, returns_{stock,t,2}, \ldots, returns_{stock,t,\text{maximum day in month}})$$

Dollar trading volume (dolvol)

$$dolvol_{stock,t} = log(vol_{stock,t-2} * cprc_{stock,t-2})$$

Share turnover (turn)

$$turn_{stock,t} = \frac{\frac{vol_{stock,t-1} + vol_{stock,t-2} + vol_{stock,t-3}}{3}}{shrout_{stock,t}}$$

Volatility of liquidity (stdturn)

$$\text{stdturn}_{stock,t} = std(\frac{vol_{stock,t,1}}{shrout_{stock,t,1}}, \frac{vol_{stock,t,2}}{shrout_{stock,t,2}}, \ldots, \frac{vol_{stock,t,\text{maximum day in month}}}{shrout_{stock,t,\text{maximum day in month}}})$$

Definitions of the 8 macroeconomic predictors are presented below.

Dividend-price ratio (dp)

$$dp = log(\frac{D12}{\text{SPX closing price at month end}}) = log(D12) - log(\text{SPX closing price at month end})$$

where D12=sum of dividends on SPX for the past 12 months (rolling sum)

SPX is an exchange-traded fund (ETF) that tracks S&P 500 index i.e. prices of various stocks are aggregated according to the S&P 500 index. S&P 500 index is a market-capitalization weighted index that constitutes the 500 largest (by market capitalization/size) publicly traded companies in the US.

Earnings-price ratio (ep)

$$ep = log(\frac{E12}{\text{SPX closing price at month end}}) = log(E12) - log(\text{SPX closing price at month end})$$

where E12=sum of earnings on S&P 500 index for the past 12 months (rolling sum)

Book-to-market ratio (b/m) is the ratio of book value to market value of the Dow Jones Industrial Average (DJIA). DJIA groups together the 30 mostly traded stocks in New York Stock

Exchange (NYSE) and similar to S&P 500, it is used as an indicator of the overall direction of stock prices and the US economy.

Net equity expansion (ntis) is the ratio of 12-month rolling sums of net issues by NYSE listed stocks to the total end-of-year market capitalization of NYSE stocks.

3-months US treasury-bill rate (tbl)
The treasury-bill rate is widely considered the benchmark for risk-free interest rate since it is backed by the US government. It represents the performance of the US economy.

Term spread (tms) = Long term government bond yield (lty) - tbl

Default spread (dfy) = AAA-rated corporate bond yield - BAA-rated corporate bond yield

Stock variance (svar) is the variance of S&P 500 daily returns for each month.

Other variables
A stock is determined uniquely by its permanent identification number (permno). Both permno and month-end dates are omitted as predictors and are only used for subsequent analysis and for splitting of data into training and test sets.

Distributions of variables

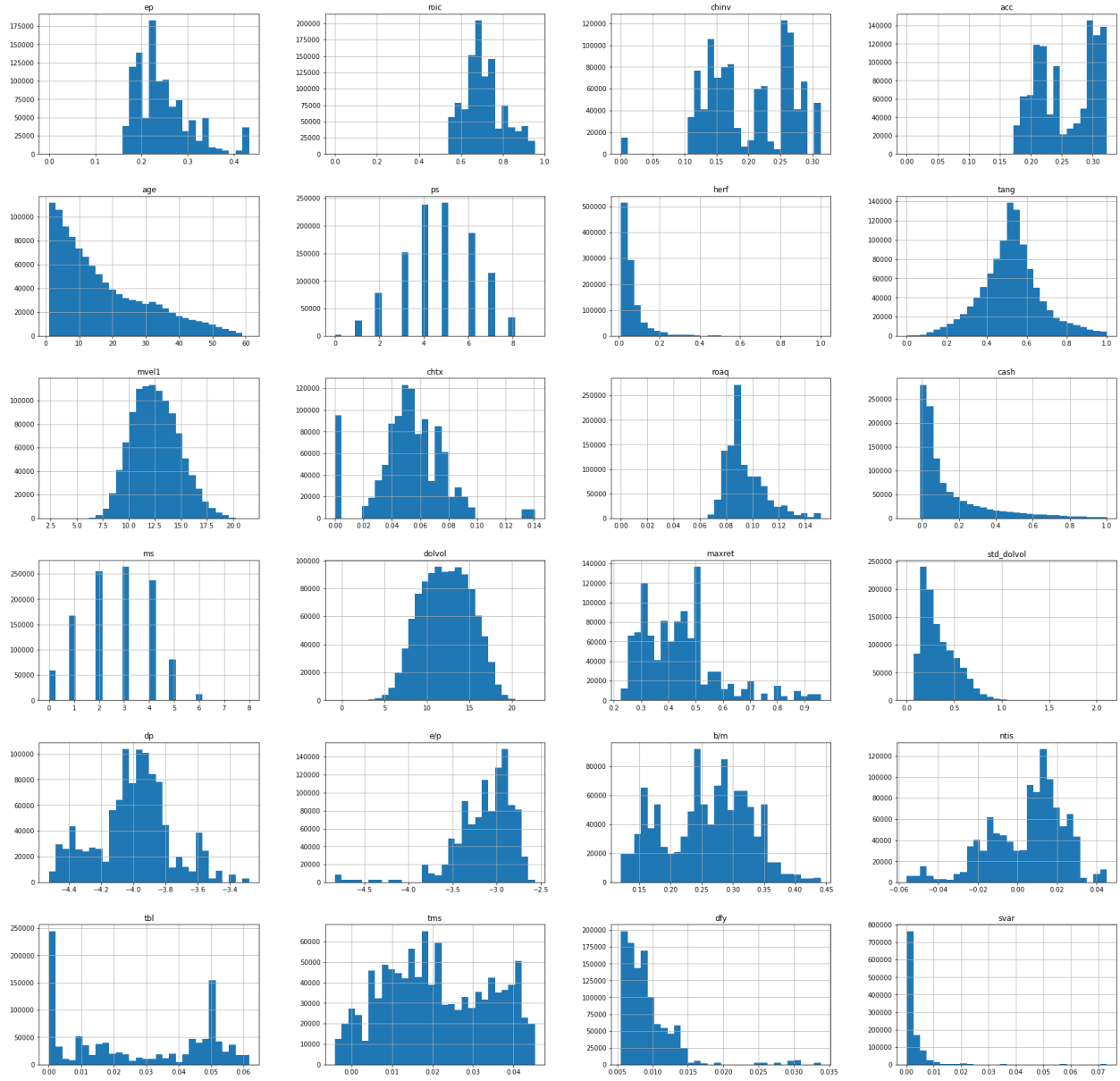

Diagram 1: Distributions of variables

Diagram 2: Distributions of variables 2

Distribution/attribute	Predictor
Truncated normal distribution	ep, roic
Gamma/weibull distribution	cash, herf, std_dolvol, age, dfy, svar
Right-skewed poisson distribution	ms
Left-skewed poisson distribution	ps
Bimodel distribution	ntis, b/m, tms, chinv, acc, dp

Numerical variables	ps, age, securedind, convind, sin, ms, nincr
Laplace distribution	tang
Left-skewed	e/p, ntis
Right-skewed	roaq, maxret

Predictors in diagram 2 are highly skewed with extreme peaks, mostly at 0. No transformation of predictors is used in this investigation.

3. Model specifications

Notations

p - no. of regressors (excluding intercept for linear regression)
n - no. of observations
X is a $n \times p$ matrix consisting of the values of all regressors
x_j is the jth column/feature of the matrix X as a column vector
$x_{(i)}$ is the ith row of the matrix X as a column vector
y_i is the response variable associated to $x_{(i)}$

Evaluation

n and samples used should be modified accordingly.

Metrics

Mean-squared error (MSE)

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Huber loss

$$\text{Huber loss} = \sum_{i=1}^{n}(I_{|y_i-\hat{y}_i|<\epsilon}\frac{(y_i - \hat{y}_i)^2}{2} + I_{|y_i-\hat{y}_i|\geq\epsilon}\epsilon(|y_i - \hat{y}_i| - 0.5\epsilon))$$

where ϵ is chosen as the 99.9 percentile of the absolute errors in this report.
Huber loss is more robust to outliers compared to MSE.

Coefficient of determination (R^2)

$$\begin{aligned} R^2 &= 1 - \frac{SSE}{SST} \\ &= 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \end{aligned}$$

In the paper, the non-demeaned version of R^2 is preferred over the original R^2.

$$\text{Modified } R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} y_i^2}$$

Splitting scheme

A train-test split is applied with training data taken from 1994 to 2002 inclusive and test data taken from 2003 and onwards (until 2020). A model is trained on the training set and used to make predictions on first-year observations in the test set. Iteratively, the first-year observations in the testing set are rolled over into the training set and first-year observations in the training set removed before being fitted to another model in the next iteration. In this way, the training set is fixed to 9 years of data. This differs from the paper where the training set is expanded

incrementally without removal of data. By fixing the duration of data in the training set, the idea is to better capture near-time features with less dilution by far-time and possibly less relevant features. On the other hand, it is noted that a large enough training set is needed to maintain a degree of generalisability. The size of the training set (9 years) is arbitrarily chosen for this report, further calibration may be needed to find a desirable balance. Altogether, 18 of such models are built to obtain predictions for the entire test set.

Assuming that the response can be expressed as a function of predictors with identically and independently distributed (iid) errors having mean 0 and constant variance.

$$y_* = f(x_{(*)}) + \epsilon_*$$

$$\sigma^2 = var(\epsilon_*) = E(\epsilon_*^2) + E(\epsilon_*)^2 = E(\epsilon_*^2)$$

Each model is represented by $\hat{f}(X)$.

$$\hat{y}_* = \hat{f}(x_{(*)})$$

Let f, $\hat{f}$ be the shorthand for $f(x_{(*)})$ and $\hat{f}(x_{(*)})$ respectively.

$$\begin{aligned}
\text{Expected prediction error(EPE)} &= E((y_* - \hat{f})^2) \\
&= E((\epsilon_* + f - \hat{f})^2) \\
&= E(\epsilon_*^2) + 2E(\epsilon_*(f - \hat{f})) + E((f - \hat{f})^2) \\
&= \sigma^2 + 2E(\epsilon_*)(f - \hat{f}) + E((f - \hat{f})^2) \\
&= \sigma^2 + E((f - \hat{f})^2) \\
&= \sigma^2 + E((f - E(\hat{f}) + E(\hat{f}) - \hat{f})^2) \\
&= \sigma^2 + E((f - E(\hat{f}))^2) + 2E((f - E(\hat{f}))(E(\hat{f}) - \hat{f})) + E((E(\hat{f}) - \hat{f})^2) \\
&= \sigma^2 + E((f - E(\hat{f}))^2) + 2E(f - E(\hat{f}))(E(\hat{f}) - E(\hat{f})) + E((E(\hat{f}) - \hat{f})^2) \\
&= \sigma^2 + Bias(\hat{f})^2 + Var(\hat{f})
\end{aligned}$$

EPE clues on the trade-off between bias and variance of predictions of a fitted model. In general, as model complexity increases, the model will be fitted more tightly to the training data (decreasing bias of training predictions) while predictions become more varied. When the decrease in bias is smaller than the increase in variance, EPE increases and overfitting occurs. Overfitting can be partially avoided by validation. A time series split divides the 9-years training set into a 8 years sub-training set and 1 year validation set to stimulate prediction on the next (test) year. This is different from the paper where an approximate 50:50 train-validation split is used for all models. The sub-training set is fitted to a model which is then used to make predictions on the validation set. This is repeated using different hyperparameters. The hyperparameters that result in the optimal loss (huber loss/MSE) on the validation set are used to build the model on the entire training set. An optimal loss is defined by a gradient condition for principal components (see methodology below) and lowest loss for all other models.

Linear regression models

1. Linear regression

Notations

j is the column of ones forming the intercept column

$$X_{reg} = [j \quad X]$$

Forms

$$y = X_{reg}\beta + \epsilon$$

$$\hat{y} = X_{reg}\hat{\beta}$$

Basic linear assumptions

$$\text{Assumption 1: } E(\epsilon) = 0$$

$$\text{Assumption 2: } var(\epsilon) = \sigma^2$$

$$\text{Assumption 3: } cov(\epsilon_i, \epsilon_j) = 0 \text{ for } i \neq j$$

Linear regression aims to minimise the l2-norm of $\hat{\epsilon}$

$$\begin{aligned}\text{Loss function} &= \hat{\epsilon}'\hat{\epsilon} \\ &= \widehat{SSE} \\ &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2\end{aligned}$$

$$\hat{\beta} = (X'_{reg}X_{reg})^{-1}X'_{reg}y \text{ (if the inverse of } X'_{reg}X_{reg} \text{ is unique)}$$

For derivation, see A1.

Features should be linearly independent for $\hat{\beta}$ to be the best linear unbiased estimator (BLUE) on X_{reg}(without any prior linear transformation). It is required that X to have full column rank for $\hat{\beta}$ to be unique (see appendix A2).

$$\text{Assumption 4: } \epsilon_i \sim N(0, \sigma^2), i = 1, \ldots, n$$

Assumption 4 is a necessary assumption for analysis using t-test and/or F-test. The derivation of t-test and F-test are described in appendix A6 and A7 respectively.

Hyperparameters

None, no validation

Variable importance

Let X_{-j} denote the matrix of X_{reg} with x_j removed.

$$\text{Partial correlation} = \hat{\rho}_{x_j y.X_{-j}}$$
$$= \frac{\sum_{i=1}^{n}(\hat{\epsilon}_{x_j \text{ on } X_{-j},i} - \bar{\hat{\epsilon}}_{x_j})(\hat{\epsilon}_{y \text{ on } X_{-j},i} - \bar{\hat{\epsilon}}_{y \text{ on } X_{-j}})}{\sqrt{\sum_{i=1}^{n}(\hat{\epsilon}_{x_j \text{ on } X_{-j},i} - \bar{\hat{\epsilon}}_{x_j \text{ on } X_{-j}})^2}\sqrt{\sum_{i=1}^{n}(\hat{\epsilon}_{y \text{ on } X_{-j},i} - \bar{\hat{\epsilon}}_{y \text{ on } X_{-j}})^2}}$$
$$= \frac{\sum_{i=1}^{n}\hat{\epsilon}_{x_j \text{ on } X_{-j},i}\hat{\epsilon}_{y \text{ on } X_{-j},i}}{\sqrt{\sum_{i=1}^{n}\hat{\epsilon}^2_{x_j \text{ on } X_{-j},i}}\sqrt{\sum_{i=1}^{n}\hat{\epsilon}^2_{y \text{ on } X_{-j},i}}}$$

Equivalently (second definition, see appendix A9 for derivation),

$$\hat{\rho}_{x_j y.X_{-j}} = \frac{t_j}{\sqrt{t_j^2 + (n-p-1)}}$$

R^2 is non-decreasing as more predictors are added into the linear regression model. Instead of fitting regression models on subsets on predictors (less the predictor) and ranking the decreases in R^2, comparing partial correlations also yields the same outcome. Obtaining the partial correlation using the second definition avoids fitting a linear regression for each feature, saving computational time. A partial correlation with a larger magnitude indicates that the associated predictor is more important to the accuracy of the model given that other predictors are present in the linear model.

$$\text{Variable importance of predictor j} = |\hat{\rho}_{x_j y.X_{-j}}|$$

2. Forward stepwise selection

Since it is computationally infeasible to build models with all combinations of predictors (2^p models), only suboptimal models are examined using forward and backward stepwise selection. Beginning with the intercept model, the variable which results in the largest decrease in SSE on its inclusion is sequentially added into the model. In this way, p+1 models of different sizes are obtained and are validated to find the model size that results in the lowest validation MSE. The linselect python library by Jonathan Landy is used in the report. The implementation differs from what is stated above but it follows a similar reasoning.

Hyperparameter
Number of predictors

Variable importance
See linear regression

3. Backward stepwise elimination

Beginning with the full model (all predictors included), the predictor which results in the smallest increase in SSE on its removal is sequentially eliminated from the model. At each step, refitting models of one size smaller to compare subsequent increases in SSE can be avoided by computing partial correlations using the second definition (see linear regression). The variable with partial correlation with the smallest magnitude is eliminated from the model. p+1 models of

different sizes are obtained and a model of the appropriate size is selected by validation. The linselect python library is used here similar to forward stepwise selection.

Hyperparameter
Number of predictors

Variable importance
See linear regression

4. **Ridge regression**

$$\text{Loss function} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

where λ is the hyperparameter

$$\hat{\beta} = (X'_{reg}X_{reg} + \lambda I)^{-1}X'_{reg}y \text{ (if the inverse of } X'_{reg}X_{reg} + \lambda I \text{ is unique)}$$

where X_{reg} is assumed to be standardised.
For derivation, see A18. Ridge regression attempts to reduce the squared l2-norm of $\hat{\beta}$ in the loss function. Decreasing $Var(\hat{f})$ while increasing bias $Bias(\hat{f})$ can result in lower EPE.

Hyperparameter
$10^{-2} \leq \lambda \leq 10^{15}$, a larger λ decreases $Var(\hat{f})$

Variable importance
Since β is required for regularisation and varies with the scale of each predictor, X_{reg} is standardised before applying to the model. After standardisation, $|\beta_j|$ can be taken directly as variable importance of the jth feature.

5. **Least absolute shrinkage and selection operator (Lasso) regression**

$$\text{Loss function} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

where λ is the hyperparameter
The solution for lasso regression can be derived using proximal gradient descent. Starting from a randomly initialised $\theta^{(0)}$, a convex function $g^{(k)}(\theta)$ is derived at each iteration k such that $g^{(k)}(\theta)$ touches $f(\theta)$ at $\theta^{(k)}$ and $g^{(k)}(\theta) \geq f(\theta) \text{ for all } \theta$. Minimising $g^{(k)}(\theta)$ lends help to the minimisation of $f(\theta)$ since g(x) can be minimised more easily, progressively bringing the x value closer to the minima of f(x). For details of the implementation, see appendix A19.
Lasso regression attempts to reduce the l1-norm of $\hat{\beta}$ in the loss function. Decreasing $Var(\hat{f})$ while increasing bias $Bias(\hat{f})$ can result in lower EPE.

Hyperparameter

$10^{-7} \leq \lambda \leq 10^5$, a larger λ decreases $Var(\hat{f})$

Variable importance

Since lasso penalty is sparse, lasso regression may perform variable selection. Importance of a variable j left out in the final model is decided by when $\beta_j^{(k)}$ first turns 0 for k>0 during gradient descent. To avoid operator bias, $\beta^{(0)}$ is initialised as the closed-form OLS solution. The variable that first exits the model is ranked as the least important. After ranking all variables with $\hat{\beta}_j = 0$, the remaining variables are ranked in a higher tier using their relative $|\beta_j|$.

6. Elastic net regression

$$\text{Loss function} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\rho\sum_{j=1}^{p}|\beta_j| + \frac{\lambda(1-\rho)}{2}\sum_{j=1}^{p}\beta_j^2$$

where $0 \leq \rho \leq 1$ is the l1 ratio and λ>0 is the shrinkage parameter

The solution for elastic net regression can be derived using proximal gradient descent similar to lasso regression. For details of the implementation, see appendix A20.

Elastic net attempts to reduce the l1 and squared l2-norm of $\hat{\beta}$ in the loss function.

Hyperparameters

$10^{-7} \leq \lambda \leq 10^5$, a larger λ decreases $Var(\hat{f})$

$0 \leq \rho \leq 1$, a larger ρ decreases $Var(\hat{f})$ to a larger extent when $\hat{\beta}_j$ are small i.e. <1, a smaller ρ decreases $Var(\hat{f})$ to a larger extent when $\hat{\beta}_j$ are large i.e. >1. l2 loss imposes a larger penalty than l1 loss when coefficients are large.

In the report, parameters are optimised singly. λ is optimised first with ρ presetted to 0.5, then ρ is optimised. This is repeated 2 times and ends with 1 last optimisation of λ.

Variable importance

Since elastic net regression performs variable selection, the variable importance is attained similar to lasso regression.

7. Smoothly clipped absolute deviation (SCAD) regression

$$J_{\lambda,a}(\beta_j) = \begin{cases} \lambda|\beta_j| \text{ if } |\beta_j| \leq \lambda \\ -\frac{\beta_j^2 - 2a\lambda|\beta_j| + \lambda^2}{2(a-1)} \text{ if } \lambda < |\beta_j| \leq a\lambda \\ \frac{(a+1)\lambda^2}{2} \text{ if } |\beta_j| > a\lambda \end{cases}$$

where $\lambda > 0$, $a \geq 1$

$$\text{Loss function} = \frac{1}{2(n-1)} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{j=1}^{p} J_{\lambda,a}(\beta_j)$$

where λ, a in $J_{\lambda,a}(\beta_j)$ are parameters.
The solution for SCAD regression can be derived using coordinate descent. The partial derivative of the loss function with respect to each beta coefficient can be obtained. Beta coefficients are updated singly while other (beta) coefficients are kept constant. The procedure terminates when the updates have looped through all coefficients and the overall change in $\beta^{(*)}$ is below a tolerance level (appendix A21 for details).
A good choice for a is 3.7 for various problems. a is initialised at this value and optimisation is done similarly to the procedure of elastic net. The difference is that every second parameter optimised is a instead of ρ. Since the SCAD loss function is non-convex, running minimisation algorithms may result in suboptimal β i.e losses stuck at local minimums. As a result, the algorithm is run multiple times with different initialisations to increase the chances of landing on a better solution for β. Initialisations are fixed in this report for reproducibility.

Initialisations

$$\beta^{(0,\text{1st initialisation})} = (X'_{reg}X_{reg})^{-1}X'_{reg}y$$
$$\beta^{(0,\text{2nd initialisation})} = (X'_{reg}X_{reg} + 0.25I)^{-1}X'_{reg}y$$
$$\beta^{(0,\text{3rd initialisation})} = (X'_{reg}X_{reg} + 0.75I)^{-1}X'_{reg}y$$

The 1st initialisation is the closed-form OLS solution, the 2nd and 3rd are variants of ridge regression solutions.

Hyperparameter

$10^{-7} \leq \lambda \leq 10^{8}$, a larger λ decreases $Var(\hat{f})$
$2.01 \leq a \leq 50$

Variable importance

Variable importance is attained similar to lasso regression.

Decision tree models

8. Decision tree regression

Definition

A terminal node is a node that has not undergone splitting. Splitting is realised in the form of a binary rule, $x_{ij} \leq a$ where a is to be determined and unique to each split. Each split is binary where observations in a node are subdivided into 2 nodes according to whether or not the binary rule is satisfied. Observations pass from the base of the tree to the leaf (or terminal nodes) separated by splits at each branch. The result is that each observation is grouped into exactly one of the terminal nodes. Altogether, there are M terminal nodes representing M regions R_m for m=1,2,...,M. The predicted response is the mean of responses in the region the observation is grouped in.

$$\begin{aligned}\text{Loss function} &= RSS \\ &= \sum_{m=1}^{M} \sum_{i \in R_m} (\hat{y}_{R_m} - y_i)^2\end{aligned}$$

where $\hat{y}_{R_m}$ is the mean of responses in region m

$$Q_m(T) = \frac{1}{n_m} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2$$

Weighted impurity decrease equation

$$\begin{aligned}\text{Decrease in impurity} &= \frac{n_t}{n}(Q_t(T) - \frac{n_{t_R}}{n_t} Q_{t_R}(T) - \frac{n_{t_L}}{n_t} Q_{t_L}(T)) \\ &= \frac{n_t}{n} Q_t(T) - \frac{n_{t_R}}{n} Q_{t_R}(T) - \frac{n_{t_L}}{n} Q_{t_L}(T)\end{aligned}$$

The decision tree starts with all observations grouped under a single node at the base of the tree. At each step, each terminal node is considered for binary splitting around every observed value of X that falls within its region domain. Across all terminal nodes, the best split is determined as the split that results in the largest decrease in impurity. The node with the best split branches out (with the best split) and the tree is grown. This process is repeated until the tree is fully-grown.

An optimised variant of a decision tree called LightGBM tree is used in this report. With the naive decision tree, a large number of observations are considered at each split which is computationally expensive. LightGBM tree considers splitting each feature into subranges or bins where number of bins << n (histogram-based sampling). LightGBM swaps out the weighted impurity decrease with Gradient-Based One Side Sampling (GOSS) in determining the optimal split. Before building a tree, negative gradients of the loss function are calculated for each observation and ranked. Gradients with larger magnitudes (top a*100% observations) are identified as under-trained samples (subset A) while gradients with smaller magnitudes (bottom (1-a)*100% observations) are identified as well-trained samples where 0<a<1. b*100%

observations (subset $\mathcal{B}$) are randomly sampled without replacement from the well-trained samples and considered for hypothetical splits in computation of $\tilde{V}$ below (0<b<1-a). Rest of the (1-a-b)*100% observations are ignored. Both histogram-based sampling and GOSS reduces the number of values to consider for each split.

$$\tilde{V}_j(s) = \frac{1}{n}\left(\frac{(\sum_{x_i \in A_{t_L}} g_i + \frac{1-a}{b}\sum_{x_i \in \mathcal{B}_{t_L}} g_i)^2}{n_{t_L}^{(j)}(s)} + \frac{(\sum_{x_i \in A_{t_R}} g_i + \frac{1-a}{b}\sum_{x_i \in \mathcal{B}_{t_R}} g_i)^2}{n_{t_R}^{(j)}(s)}\right)$$

where g_i are negative gradients for $1 \leq i \leq n$, and

$$A_{t_L} = \{x_i \in A, x_{ij} \leq s\}, A_{t_R} = \{x_i \in A, x_{ij} > s\}, \mathcal{B}_{t_L} = \{x_i \in \mathcal{B}, x_{ij} \leq s\}, \mathcal{B}_{t_R} = \{x_i \in \mathcal{B}, x_{ij} > s\}$$

The optimal split s is the split that results in the largest variance gain $\tilde{V}$. In this report, the growth of the tree is bounded by the maximum number of terminal nodes $|T|_{max}$.

Hyperparameter

$1 \leq |T| \leq 16$, a larger $|T|$ increases $Var(\hat{f})$

Variable importance

A larger variance gain after a split in a feature range suggests that the particular feature contributes more to a lower RSS of the overall tree. Variance gains are summed across all splits for each feature and treated as the variable importance.

9. Bagging

In bagging, B decision trees are each grown with a bootstrap dataset of size n. Each bootstrap dataset is generated by randomly sampling n observations from the original dataset with replacement.

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}_b(x)$$

where B is the number of decision trees used in bagging

$$\text{P(ith observation not the first sample of bootstrap dataset)} = 1 - \frac{1}{n}$$

$$\begin{aligned}\text{P(ith observation not included in bootstrap dataset)} &= (1-\frac{1}{n})^n \\ &= e^{-1} \text{ as n} \to \infty \\ &= 0.36... \\ &\approx \frac{1}{3}\end{aligned}$$

When n is large, the probability that an observation is included in a bootstrap dataset is ~0.63. The 'bagging_fraction' parameter is set for LightGBM to simulate bagging.

When B is large, $Var(\hat{f}_{bag})$ stabilises. The training and validation errors tend to level off when a sufficiently large number of trees is used. The idea of bagging follows the concept of bias-variance tradeoff where the predictions of each tree is biased for the bootstrapped dataset the tree is fitted to, potentially increasing $Bias(\hat{f})$ while a lower $Var(\hat{f})$ is attained by averaging predictions. When decrease in $Var(\hat{f})$ is more than increase in $Bias(\hat{f})^2$, bagging is preferred over a single decision tree for lower EPE.

Hyperparameter

$1 \leq d \leq 6$, a larger depth (d) increases $Var(\hat{f})$
Sufficient large B, $1 \leq B \leq 300$

Variable importance

Variable importance is calculated as the sum of variable importance of individual trees.

10. Random forest (RF)

Random adds on to the bagging with the concept of decorrelating trees by allowing a random subset of m < p predictors to be considered for each split. A common value for m is $\lfloor \frac{p}{3} \rfloor$ for regression which is also used in this report. Decorrelation reduces the chance of strong predictors masking the effects of weaker predictors.

$$\begin{aligned}Var(\hat{f}_{bag}) &= Var(\frac{1}{B}\sum_{b=1}^{B}\hat{f}_b) \\ &= \frac{1}{B^2}\sum_{b=1}^{B}Var(\hat{f}_b) + \frac{2}{B^2}\sum_{1\leq b_1<b_2\leq B}Cov(\hat{f}_{b_1},\hat{f}_{b_2})\end{aligned}$$

Loosely, covariance terms in the context of bagging are positive i.e. predictions for each bagged tree should be similar (in direction). Decorrelating trees has an probable effect of deflating the covariances in the summation above and leads to a reduction in $Var(\hat{f})$. Similar to bagging, a reasonably large number of trees (B) is required before changes in $Var(\hat{f})$ are minimal.

Hyperparameters
Same as bagging

Variable importance
Variable importance can be calculated similar to bagging.

11. Gradient-boosted random trees (GBRT)

Procedure

Step 1: Initialise $\hat{f}_{boost}(X) = 0$ and r=y

Step 2: For b=1,2,...,B

(a) Fit a tree $\hat{f}_b$ with d splits

(b) $\hat{f}_{boost}(X) \leftarrow \hat{f}_{boost}(X) + \lambda \hat{f}_b(X)$

(c) $r \leftarrow r - \lambda \hat{f}_b(X)$

Step 3: $\hat{f}_{boost}(X) = \sum_{b=1}^{B} \lambda \hat{f}_b(X)$

where B is the number of trees used in boosting and λ is the shrinking parameter/learning rate

During each iteration of boosting, the predictions of a fitted decision tree is shrunk by a factor of λ to allow for newer trees a chance to tackle residuals. Individual trees are grown as stumps, having a maximum of d splits or maximum depth of 2-3. $Var(\hat{f})$ is controlled by having stumps with small individual $Var(\hat{f}_b)$ and lowered pairwise correlation since a newer tree is decorrelated by fitting on residuals from predictions of previous trees. At the start of each boosting iteration, the negative gradients are updated for each lightGBM tree.

λ is commonly set to 0.01 or 0.001. λ is set to 0.01 in this case. Unlike boosting and random forest, overfitting will occur when B is too large. A smaller λ will require a larger B.

Hyperparameters

$1 \leq d \leq 3$, a larger depth (d) increases $Var(\hat{f})$

$1 \leq B \leq 1000$

Variable importance
Variable importance can be calculated similar to bagging.

Dimensional reduction

12. Principal Component Regression (PCR)

X is assumed to be standardised. Standardisation is required since PCR uses projected variances of predictors to construct the principal components (more information below).
Form

$$\Sigma e_j = \lambda_j e_j$$

where each (e_j,λ_j) represents an eigenpair of Σ for i=1,2,...,p
Each principal component z_j is constructed using the e_j as weights/loadings i.e. $z_j = Xe_j$

$$\begin{aligned} var(z_j) &= var(Xe_j) \\ &= e_j^{'}\Sigma e_j \\ &= e_j^{'}\lambda_i e_j \text{ (due to definition of eigenvectors)} \\ &= \lambda_j e_j^{'} e_j \\ &= \lambda_j \end{aligned}$$

$$\begin{aligned} cov(z_j, z_k) &= cov(Xe_j, Xe_k) \\ &= e_j^{'}\Sigma e_k \\ &= \lambda_i e_j^{'} e_k \\ &= 0 \end{aligned}$$

Each z_j is ranked according to its corresponding eigenvalue λ_j which coincides with its variance. Visually, the first principal component (with the highest λ_j) represents the direction along which the projected observations vary most. Subsequent components are constructed likewise constrained on orthogonality with previous components. Since z_j are pairwise uncorrelated and jointly maximise variance being explained, z_j are good candidates to compactly capture variance in the predictors.

$$\begin{aligned} \text{Proportion of variance explained by kth component} &= \frac{var(z_k)}{trace(\Sigma)} \\ &= \frac{var(z_k)}{trace(PDP^{-1})} \\ &= \frac{var(z_k)}{trace(DP^{-1}P)} \\ &= \frac{var(z_k)}{trace(D)} \\ &= \frac{\lambda_k}{\sum_{i=1}^{p}\lambda_i} \end{aligned}$$

where $\Sigma = PDP^{-1}$ is the diagonalized form of Σ
The formation of principal components is independent from the response. The principal components capture variation in the predictors and the hope is that these directions can aid in predicting the response in a linear regression model.

Hyperparameter

No. of principal components to use for regression, $1 \leq M \leq p$

Variable importance

Since the predictors are demeaned (standardised),

$$z_j = \sum_{i=1}^{p} e_{ij} x_j$$

$$\bar{z}_j = \sum_{i=1}^{p} e_{ij} \bar{x}_j$$

$$\bar{z}_j = \sum_{i=1}^{p} e_{ij}(0) = 0$$

$$\text{Standardised } z_j = \frac{1}{\sqrt{\lambda_j}} z_j$$

$$\begin{aligned}
\hat{y} &= \hat{\beta}_0 j + \sum_{j=1}^{m} \hat{\beta}_j (\frac{1}{\sqrt{\lambda_j}} z_j) \\
&= \hat{\beta}_0 j + \sum_{j=1}^{m} \hat{\beta}_j X \frac{1}{\sqrt{\lambda_j}} e_j \\
&= \hat{\beta}_0 j + \sum_{j=1}^{m} \hat{\beta}_j \sum_{k=1}^{p} x_k \frac{e_{kj}}{\sqrt{\lambda_j}} \\
&= \hat{\beta}_0 j + \sum_{j=1}^{m} \hat{\beta}_j \sum_{k=1}^{p} x_k \frac{e_{kj}}{\sqrt{\lambda_j}} \\
&= \hat{\beta}_0 j + \sum_{k=1}^{p} (\sum_{j=1}^{m} \hat{\beta}_j \frac{e_{kj}}{\sqrt{\lambda_j}}) x_k
\end{aligned}$$

Decomposing the terms in the linear regression, $|\sum_{j=1}^{m} \hat{\beta}_j \frac{e_{kj}}{\sqrt{\lambda_j}}|$ is obtained as the importance of the kth feature.

$|\sum_{j=1}^{m} \hat{\beta}_j \frac{e_{kj}}{\sqrt{\lambda_j}}|$where e_{kj} is the coefficient of the jth component associated to the kth predictor

13. Partial Least Squares (PLS)

Procedure

Step 1. Standardise/demean X

Step 2. Set $x_j^{(0)} = x_j, j = 1, ..., p$

Step 3. For m=1,...,M

(a) Compute $\phi_{mj} = \langle x_j^{(m-1)}, y \rangle$ for each j

(b) Construct $z_m = \sum_{j=1}^{p} \phi_{mj} x_j^{(m-1)}$

(c) $x_j^{(m)} = x_j^{(m-1)} - \frac{\langle z_m, x_j^{(m-1)} \rangle}{\langle z_m, z_m \rangle} z_m$

Step 4. $\hat{y} = \bar{y}1 + \sum_{m=1}^{M} \frac{\langle z_m, y \rangle}{\langle z_m, z_m \rangle} z_m$

The information explained by prior PLS components are removed from predictors i.e. new predictors at each iteration are residuals from the regression of individual predictors on all prior principal components. At step 3c, since all prior PLS components except the latest derived (i.e. z_i for i=1,2,...t) are orthogonal to the predictors $x_j^{(t)}$, the regression coefficients corresponding to these orthogonal PLS components are all zeroes. Therefore, it is only required for regression to be performed on the latest derived principal component to obtain the new residuals. (See appendix B3 for proof)

Hyperparameter

Number of components to use for regression, $1 \leq M \leq p$

Components enter in order of $z_1, z_2, ...z_M$

Variable importance

Variable importance in projection (VIP)

$$VIP_j = \sqrt{n \frac{\sum_{m=1}^{M} \hat{\beta}_m^2 z_m' z_m (\frac{\phi_{mj}}{\|\phi_m\|})^2}{\sum_{m=1}^{M} \hat{\beta}_m^2 z_m' z_m}} \text{ where } \hat{\beta}_m = \frac{\langle z_m, y \rangle}{\langle z_m, z_m \rangle}$$

14. Sliced Inverse Regression (SIR)

Procedure

Step 1: Standardise X, $Z = \hat{\Sigma}^{-\frac{1}{2}}(X - j\bar{x}')$

Step 2: Divide the range of response variable into H slices without any overlap, $l_1, l_2, ..., l_H$

$$\hat{p}_h = \frac{1}{n} \sum_{i=1}^{n} 1_{\{y_i \in l_h\}}$$

Step 3: Compute $E(Z|Y_i \in l_h) = \hat{m}_{(h)}$ for each slice i.e. h=1,2,...,H

$$\hat{m}_{(h)} = \frac{1}{n\hat{p}_h} \sum_{y_i \in l_h} z_{(i)} \text{ where } \hat{m}_{(1)}, \hat{m}_{(2)}, \dots \hat{m}_{(H)} \text{ are the rows of M}$$

Step 4: Diagonalize weighted covariance matrix of E(Z|Y), $\hat{V}$ (weighted by the number of observations in each slice, mean of predictors taken to be 0 since Z is standardised), the eigenvectors are $\hat{\eta}_1, \hat{\eta}_2, ...\hat{\eta}_H$

$$\hat{V} = \sum_{h=1}^{H} \hat{p}_h \hat{m}_{(h)} \hat{m}'_{(h)}$$

Step 5: Choose K eigenvectors with the largest eigenvalues (compact representation of variation in the subspace). Output $\hat{\zeta}_k = \hat{\Sigma}^{-\frac{1}{2}} \hat{\eta}_k$

Since each $\hat{m}_{(h)}$ is a conditional expectation of observations of Z given a range of y i.e. M is Z compacted using averages across each slice, $\hat{\eta}_1, \hat{\eta}_2, ...\hat{\eta}_H$ which are the principal components of M can similarly be used to approximate the principal components of Z. More precisely, an assumption of SIR is that the distribution of X is elliptically symmetric (e.g. normal or student-t distribution). (see appendix B4)

By setting $\hat{\eta}_k = \hat{\Sigma}^{\frac{1}{2}} \hat{\zeta}_k$,

$$\begin{aligned}
\hat{\eta}_k Z_{(i)} &= \hat{\zeta}'_k (\hat{\Sigma}^{\frac{1}{2}})' \hat{\Sigma}^{-\frac{1}{2}} (x_{(i)} - \bar{x}) \\
&= \hat{\zeta}'_k (x_{(i)} - \bar{x}) \\
&= \hat{\zeta}'_k x_{(i)} + \hat{\beta}'_k \bar{x} \\
&= \hat{\zeta}'_k x_{(i)} + c_k
\end{aligned}$$

$\hat{\eta}_k Z_{(i)}$ differs from $\hat{\zeta}'_k x_{(i)}$ by a constant c_k (independent of i). c_k terms can be ignored since these terms, if included, will be pooled into the intercept term and do not value-add to the accuracy of the final regression model.

Hyperparameter

Number of components $1 \leq K \leq p$

Variable importance

Permutation importance is a model agnostic procedure for variable importance. Permutation importance randomly shuffles a predictor column and determines the average change in prediction accuracy (R^2 in this report) over a number of iterations. In contrast to refitting the model to a reduced set of variables and comparing performances, permutation importance benefits from computational savings since the procedure can be applied on the original model. Mixing up the predictor column breaks the relationship between the response and the predictor while maintaining values within feature range. By losing the predictive ability of a feature, the predictive R^2 is expected to drop and decrease attributed to the importance of the predictor. In SIR, prior to transformation with $\hat{\zeta}_k$, feature column are shuffled one at a time and (decreases in) R^2 are compared between final linear regressions of components.

Neural networks

15. Neural network

The report uses 3 model architectures (except the smallest and largest neural network) explored in the paper. Each neural network consists of hidden layer/s of dense units sandwiched between an input (topmost) layer and an output (bottommost) layer. The input layer consists of p units where each feature is separately fed into while the output layer is a dense unit where individual predictions are churned out. 2-4 hidden layers (so named NN1 to NN4) are considered. Only adjacent layers are connected and the number of units in a hidden layer is progressively halved from top to down in an inverted pyramidal manner. The topmost hidden layer is fixed at 32 hidden units.

Rectified linear unit (ReLU)

$$ReLU(x) = \begin{cases} \text{max\_value if } x \geq \text{max\_}\mathit{value} \\ x \text{ if threshold} \leq x < \text{max\_}\mathit{value} \\ \text{negative\_}\mathit{slope} * (x - \text{threshold}) \text{ otherwise} \end{cases}$$

A ReLU activation with no maximum value, threshold of 0 and negative slope of 0 is used for all dense units. This simplifies the ReLU(x) to max(0,x).

Formulation

$$x_{1,q} = x_q$$

$$x_{t,q} = ReLU(\theta_{t,q,0} + \sum_{j=1}^{n_{t-1}} \theta_{t,q,j} x_{t-1,j})$$

where $x_{t,q}$ is the output of the q_{th} unit of the t_{th} layer (numbering includes input and output layer), $\theta_{t,q}$ are weights (including bias) of the q_{th} (dense) unit of the t_{th} layer and n_t is the number of units in the t_{th} layer

Training is done using mini-batch gradient descent (MGD). The training set is divided into similar-sized sets called batches. Each batch is fed through the neural network and loss of the entire batch is computed and back-propagated. Partial derivatives of loss with respect to each parameter $\nabla_\theta L(\theta)$ are calculated using product rule. Partial derivatives of parameters held by units closer to the output unit are computed earlier and are recursively multiplied out, giving the notion of back-propagation. At the extremes, stochastic gradient descent (SGD) updates parameters after each observation while batch gradient descent (BGD) updates parameters after every epoch. Each epoch constitutes an entire training set being passed into the network. MGD reduces the variation of each update in favour for faster convergence than SGD while retaining some variation to have better chances than BGD of jumping into a better local-minima in a non-convex loss surface. In the paper, the term SGD used refers to MGD. Commonly, SGD and MGD are used interchangeably.

Learning rate shrinkage is done using an Adaptive moment estimation (ADAM) optimiser.
Let η denote the learning rate.
The basic form of an optimizer is momentum.

Momentum procedure

Step 1: Initialise $\theta^{(0)}$ randomly. Set $v^{(0)} = 0, t = 0$.

Step 2. While training loss not converged,

(a) $t \leftarrow t + 1$

(b) $g^{(t)} \leftarrow \nabla_\theta L(\theta^{(t-1)})$

(c) $v^{(t)} \leftarrow \gamma v^{(t-1)} + \eta g^{(t)}$

(d) $\theta^{(t)} \leftarrow \theta^{(t-1)} - v^{(t)}$

$\nabla_\theta L(\theta^{(t-1)})$ contains the infinitesimal changes in loss with respect to each parameter. A small value (usually $\leq$0.01) is chosen for η, representing the maximal acceptable step size where the infinitesimal gradients are believed to not change drastically e.g. gradients become positive. A larger η accelerates training but increases tendency to overshoot local minima especially in a loss function with negative spikes. γ is the proportion of momentum preserved from the previous iteration.

ADAM procedure

Step 1: Initialise $\theta^{(0)}$ randomly. Set $m^{(0)} = 0, v^{(0)} = 0, t = 0$

Step 2. While training loss not converged,

(a) $t \leftarrow t + 1$

(b) $g^{(t)} \leftarrow \nabla_\theta L(\theta^{(t-1)})$

(c) $m^{(t)} \leftarrow \beta_1 m^{(t-1)} + (1 - \beta_1) g^{(t)}$

(d) $v^{(t)} \leftarrow \beta_2 v^{(t-1)} + (1 - \beta_2) g^{(t)} \circ g^{(t)}$ where $\circ$ represent the pointwise (Hadamard) product

(e) $\hat{m}^{(t)} \leftarrow \dfrac{m^{(t)}}{(1 - \beta_1^t)}$

(f) $\hat{v}^{(t)} \leftarrow \dfrac{v^{(t)}}{(1 - \beta_2^t)}$

(g) $\theta^{(t)} \leftarrow \theta^{(t-1)} - \eta \hat{m}^{(t)} \oslash (\sqrt{\hat{v}^{(t)}} + \epsilon)$ where $\oslash$ represents the pointwise (Hadamard) division

The default is $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$. Only η needs to be tuned in most cases. η is fixed at 0.01 in this report.

ADAM optimiser extends momentum by using exponentially smoothed first and second moments (in 2(c) and 2(d)). Smoothed moments are bias corrected in 2(e-f) and in 2(g), the 'standardised' momentum is shrunk and used as an update. ϵ is an arbitrary small number to avoid division by 0.

Early stopping is a strategy that assumes that updates are stuck at a local minimum and terminates training (to save computational cost) when validation error does not decrease (below the optimal loss) within a fixed number of epochs. On termination, the model reverts to the set of parameters corresponding to the lowest validation loss.

Batch normalisation (BN)

$$\mu_{mov} \leftarrow \alpha * \mu_{mov} + (1-\alpha)\mu_{batch}$$

$$\sigma^2_{mov} \leftarrow \alpha * \sigma^2_{mov} + (1-\alpha)\sigma^2_{batch}$$

Each batch is standardised with μ_{mov} and σ^2_{mov} after application of an activation function. BN stabilises the representative power of output of each dense unit, reducing the chances of exploding or diminishing effects especially in larger networks. This potentially also gives rise to faster convergence.

Hyperparameter
λ takes 10 equally spaced values in [0.001,0.01]
λ is the constant l1 penalty (for weights) applied across all cells.

Variable importance
Variable importance used is described in Olden et al. 2004. Each predictor is identified by the input unit its values are fed into. Importance of a route is calculated as the (absolute) product of all connection weights along the route. Importance of a predictor is then the summation of importances of all routes linking its associated input unit and output unit. Equivalently, the feature importance is exactly the absolute value of the output of a reduced form of neural network without biases, normalisation and activations with input of 1 on the particular feature and 0 on other features.