Forecast Error & Loss Function

Forecast Error : $e = Y - \hat{Y}$
Squared Loss : $L(e) = (Y - \hat{Y})^2$
Absolute Loss : $L(e) = |Y - \hat{Y}|$
Properties of loss functions :
1.$L(0) = 0$
2.$L(e) \geq 0, \forall e$
3.Non-increasing if $e < 0$
4.Non-decreasing if $e > 0$
5. Risk : $R(Y) = E(L(e)) = E(L(Y - \hat{Y}))$

Bias Variance Trade Off

$E[(y - \hat{f}(x))^2] = Var(\hat{f}(x)) + [Bias(\hat{f}(x))]^2 + Var(\epsilon)$

Validation Set Approach

1.Split data into 2 sets (train and test)
2.Train the data and test it on the test set
3.Compute the MSE
Cons :
1.loss of data $\Rightarrow$ Higher MSE
2.estimate of the test error is highly variable

K-fold cross-validation

1.Split the data into K fold
2.For each degree, compute MSE for each k block
3.Average each MSE for each degree
1.$CV_{(K)} = \frac{1}{K}\sum_{k=1}^{K} MSE_k$
2.$\hat{Var}(MSE_k) = \frac{1}{K-1}\sum_{k=1}^{K}(MSE_k - M\bar{S}E_k)^2$
3.$\hat{SE}(CV_{(K)} = \sqrt{\frac{1}{K}\hat{Var}(MSE_k)}$
4.$CV_{(K)} = \frac{1}{K}\frac{1}{n_k}\sum I(y_i \neq \hat{y}_i)$

Some Useful formulae

1. $RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
2. $C_p = \frac{1}{n}(RSS + 2k\hat{\sigma}^2)$
3. Adjusted $R^2 = 1 - \frac{n-1}{n-k-1}\frac{RSS}{TSS}$
4. $AIC = \frac{1}{N\hat{\sigma}^2}(RSS + 2k\hat{\sigma}^2)$
5. $BIC = \frac{1}{N\hat{\sigma}^2}(RSS + log(N)k\hat{\sigma}^2)$

Best Subset Selection

1.Let $\bar{p} \leq p$
2.For each $k \in 0, ..., \bar{p}$, fit exactly k predictors to the model
3.Pick the highest R^2 model for each k
4.Choose the best among these selected models using AIC, BIC, CV
Cons :
1.High computational time, 2^p possible models
2.The prediction is highly unstable
3.easy to overfit if P is large

Methods when applying IC

1.First approach : Use $\hat{\sigma}^2$ directly from each model
2.Second approach : When $P << N$, use the $\hat{\sigma}^2$ from the largest model
3.Third approach : When $\frac{P}{N}$ is not small, use iterative procedure
a. use $Var(Y)$ as σ^2 and select best model based on AIC/BIC
b. from this selected model, take its σ^2 and select best model based on AIC/BIC
c. repeat this procedure until it converges

Forward Stepwise Selection

1.Starts from 1 regressor, fit all possible regressor to the model
2.Select the model with the highest Adjusted-R^2
3.Add regressors to the selected model, choose the one with the highest R^2
4.Repeat the procedure
5.Choose the best model among the selected models for each degree using AIC/BIC/CV
6. Pros and Cons:
a. P : Can be applied in high-dimensional scenarios, where $n < p, \bar{p} < n$
b. P : There are only $1 + \frac{p(p+1)}{2}$ possible models for each p
c. C : May not always find the best possible model

Backward Stepwise Selection

1. Exclude 1 regressor from the full model, choose the one with highest R^2 (exclude the one with highest $p - value$)
2. Exclude 1 regressor from the selected model, choose the one with highest R^2 (exclude the one with highest $p - value$)
3. Repeat the procedure
4. Choose the best among the selected model usign AIC/BIC/R^2
5. Pros and Cons :
a. P : Only p + 1 possible models for each p
b. C : Constrain the search space to reduce variance, incur more bias
c. C : Can't be use when P > N

Ridge Regression

1. Incur some bias but greatly reduce variance
2. $\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{P}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{P}\beta_j^2$
3. $||\beta||_2 = \sqrt{\sum_{j=1}^{P}\beta_j^2}$
4. β is not scale invariant, so we need to standardize variables before fitting them to the model
5. $\hat{\beta}_\lambda^R = \frac{\hat{\beta}_{OLS}}{(1+\lambda)}$
6. $E(\hat{\beta}_\lambda^R) = \frac{E(\hat{\beta}_{OLS})}{(1+\lambda)} = \frac{\beta}{(1+\lambda)}$
7. $Var(\hat{\beta}_\lambda^R) = \frac{Var(\hat{\beta}_{OLS})}{(1+\lambda)}$
8. Estimators are biased but has lower variance than OLS
9. Pros and Cons :
a. P : Can be used when P > N
b. P : Works best when some features are correlated
c. C : Does not select the model simultaneously

How to choose λ

1. Run the Ridge regression
2. Test different values for λ, use it to do CV
3. Choose the value of λ that minimizes MSE
4. higher λ, higher bias, lower variance

LASSO Regression

1. Shrink and select coefficients
2. $\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{P}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{P}|\beta_j|$
3. $||\beta||_1 = \sum_{j=1}^{P}|\beta_j|$
4. Estimators are not scale invariant, need to standardize variables first
5. doesn't have closed form solution
6. $\hat{\beta}_j^L(\lambda) = \hat{\beta}_j^{OLS} - \lambda/2$ if $\hat{\beta}_j^{OLS} > \lambda/2$
7. $\hat{\beta}_j^L(\lambda) = \hat{\beta}_j^{OLS} + \lambda/2$ if $\hat{\beta}_j^{OLS} < -\lambda/2$
8. $\hat{\beta}_j^L(\lambda) = 0$ if $|\hat{\beta}_j^{OLS}| \leq -\lambda/2$
9. Pros and Cons :
a. P : Helps to select variables
b. P : Works even if $P >> N$
c. C : Tends to select only one of the correlated features
d. C : Can't use LOOCV because there is no closed form solution

How to choose λ

1. Run CV many times and chosoe value of λ that occurs the most
2. 1SE rule : Calculate the SE fo the estimated MSE for eachλ on the grid and then select the largest λ for which the estimated CV MSE is within 1 SE of the minimum-MSE λ choice
3. Higher $\lambda \Rightarrow$ lower penalization $\Rightarrow$ lower model complexity
4. Plug-in λ : $\lambda = 2c\sqrt{n}\hat{\sigma}\Phi^{-1}(1 - \gamma/2p)$, c = 1.1 is suggested
5. Although LASSO selects variables, the coefficients are still biased
6. Sparsity (Some coefficient is 0 and non-zero coefficient is large) is assumed but may not be the case in reality

Post-LASSO Estimation

1. Run LASSO regression
2. Use the selected variables and run OLS again
3. Works well with theoretically motivated plug-in penalty (c = 1.1 is suggested)
4. Sure Independence Screening (SIS) : rank variables by marginal correlations and only consider d most highly correlated to Y variables (i.e. run univariate regression on Y for each variable and choose d most correlated variables)
5. $d = N - 1$ or $d = N/log(N)$ is recommended

Principal Components

1. Consider linear combinations $Z_1,\ Z_2, \ldots,\ Z_M$ of the original $\boldsymbol{X}$
2. $Z_m = \sum_{j=1}^{P}\phi_{jm}X_j$
3. $y_i = \theta_0 + \sum_{m=1}^{M}\theta_m z_{im} + \epsilon_i$
4. $\boldsymbol{\phi_1} = argmax_{\phi\phi'=1} var[\boldsymbol{X\phi}]$
5. where $\boldsymbol{\phi_1}$ is called loading/weight. What we are doing here is to find unit vectors that maximize $\boldsymbol{var(X\phi)}$, while they are orthogonal to each other.
6. PC, Z, is the linear combination of $\boldsymbol{X}$, where the coefficients are ϕ, ranked by contribution to variance.
7. ϕ are the eigenvectors of the covariance matrix $\Sigma = var[\boldsymbol{X}]$
8.
$var\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix}$ where $\sigma_1^2 + \sigma_2^2 = \sigma_x^2$
9. An eigenvector $u_j = (u_{j1} \quad u_{j2})'$ satisfied $\boldsymbol{\Sigma u_j} = D_j\boldsymbol{u_j}$
10. Eigenvectors are orthogonal to each other (ie. $\boldsymbol{u_1} \cdot \boldsymbol{u_2} = 0$)
11. $\|u_1\| = \sqrt{u_{11}^2 + u_{12}^2}$
12. $\boldsymbol{Z_i} = \boldsymbol{X\phi_1}$, where ϕ_i is a unit eigenvector, and $\boldsymbol{Z}$ is a PC.
13. $var(\boldsymbol{Z_i}) = D_i$, where D_i is $\boldsymbol{\phi_i}$'s associated eigenvalue
14. $\boldsymbol{Z}$ is the linear combination of $\boldsymbol{X}$ using eigenvectors as weights
15. $\boldsymbol{X}$ can be expressed as a linear combination of $\boldsymbol{Z}$
16. $\boldsymbol{X = Z\Phi^{-1} = Z\Phi'}$, since Φ is orthogonal matrix.
17. The ratio that gives the share of total variation in the data explained by each PC (Z_i) is $\frac{D_i}{\sum_{i=1}^{P}D_i}$
18. $\boldsymbol{\Sigma} = (\boldsymbol{\phi_1} \quad \boldsymbol{\phi_2}) \times \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix} \times \begin{pmatrix} \boldsymbol{\phi_1'} \\ \boldsymbol{\phi_2'} \end{pmatrix} = D_1\boldsymbol{\phi_1\phi_1'} + D_2\boldsymbol{\phi_2\phi_2'}$
19. PC are not scale-invariant, we need to standardize the data
20. PCA works best on highly correlated data. If $\boldsymbol{X}$ are uncorrelated, no dimension reduction via PCA is possible

PCR

1. Run PCA on the predictor matrix $\boldsymbol{X}$
2. Construct the sample principal components $\boldsymbol{Z}$
3. Remove all but $M < P$ first PC
4. Regress $\boldsymbol{y}$ on $\boldsymbol{z_1, \ldots, z_M}$
1. It can be shown that Ridge shrinks the coefficients of PC by $\frac{D_j}{D_j+\lambda}$
2. Thus, PC corresponding to lowest variance directions get shrunk the most
3. Assumption: the directions of highest variability in $\boldsymbol{X}$ are those most associated with $\boldsymbol{y}$, but this may not be the case
4. There is no variable selection in PCR, all $\boldsymbol{X}$ are used to form PC

PLS

1. PLS finds directions helpful in explaining both $\boldsymbol{Y}$ and $\boldsymbol{X}$
2. To find the dicrection $\boldsymbol{Z_1}$, regress $\boldsymbol{Y}$ on each individual $\boldsymbol{X_j}$ and set the weights ϕ_{1j} to the respective OLS coefficients
3. We estimate $y_i = \phi_{1p}x_{ip} + \epsilon_i,\ \forall\ p$
4. Then construct $Z_1 = \sum_{j=1}^{P}\hat{\phi}_{1j}X_j$, this will be our first PC
5. After that, $x_{ip} = \beta_p Z_{1i} + u_{pi},\ \forall\ p$
6. $y_i = \phi_{2p}\hat{u}_{ip} + \epsilon_i,\ \forall\ p$
7. Then the second PC is $Z_2 = \sum_{j=1}^{P}\hat{\phi}_{2j}\boldsymbol{\hat{u}}_j$
8. Repeat the steps above

Regression Trees

1. Regression trees splits the covariate space into a set of rectangles and then fit a simple model in each one.
2. $f(x) = \sum_{m=1}^{M} c_m\ I(x \in R_m)$
3. $\hat{c}_m = ave(y_i|x_i \in R_m)$
4. Ideally, we want to minize $\sum_{m=1}^{M}\sum_{i\in R_m}(y_i - \hat{y}_{R_m})^2$, but this is not possible
5. STEP 1: define $R_1(j,s) = X|X_j < s$ and $R_2(j,s) = X|X_j \geq s$ and choose j and s as: $argmin\sum_{i:x_i\in R_1(j,s)}(y_i - \hat{y}_{i,R_1})^2 + \sum_{i:x_i\in R_2(j,s)}(y_i - \hat{y}_{i,R_2})^2$
6. We do argmin for each dimension, and for each dimension, do argmin to find s.
7. The set of leaves gives a partition of the X space
8. Continuous data, the prediction will be the sample mean of Y in that node
9. Categorical data, the prediction will be the majority in that node
10. Trees handle both categorical and numeric X and Y well. We no need to worry about the scale of the X. It is computationally efficient. Small trees are very interpretable. Performs automatic interaction detection. Does variable selection.
11. Does not handle categorical variables when there are a lot of categories. Deep trees lose interpretability. Unstable with respect to training data.
12. Tree choice can be represented as a minimization problem:
$\hat{T}_\alpha = arg\ min_T\ Q(T,y) = arg\ min_T\ L(T,y) + \alpha|T|$
13. In quadratic loss,
$\hat{T}_\alpha = arg\ min_T\ \sum_{m=1}^{T}\sum_{i:x_i\in R_m}(y_i - \hat{y}_{R_m})^2 + \alpha|T|$
14. α is the penalty parameter. If $\alpha = 0$, we would choose the tree that perfectly fits the data resulting in overfitting.

Classification Trees

1. Loss function: $L(T,y) = \sum_{m=1}^{T}N_m L_m(T, y_m)$, m indexes terminal nodes in tree T. N_m is the no of obs. in terminal node m. y_m is the vector of outcomes for obs. in terminal node m.
2. $\hat{p}_{mk} = \frac{1}{N_m}\sum_{i:x_i\in R_m}1(y_i = k)$,
$k(m) = arg\ max_k\ \hat{p}_{mk}$
3. Misclassification error:

$\frac{1}{N_m}\sum_{i \in R_m} 1(y_i \neq k(m)) = 1 - \hat{p}_{mk}(m)$ 4. Gini index:
$\sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk})$
completely pure $\Rightarrow$ Gini index = 0
5. Cross-entropy/deviance:
$-\sum_{k=1}^{K} \hat{p}_{mk} log(\hat{p}_{mk})$
6. We use gini index or deviance to grow the tree, and use misclassificaiton error to prune the tree.

Performance Measures

1. Lift = $\frac{TP}{TP+FP}$, within positive, how many are correct.
2. ROC = $\frac{TP}{FP}$, we want to maximize AUC.
3. Both lift and ROC consider performance of $\hat{p}(x)$ by looking at predictions $\hat{y} = 1(\hat{p}(x) > s)$ as s varies
4. $s = 1 \Rightarrow 1(\hat{p}(x)) = 0 \Rightarrow \hat{y}$ predicts all negative
4. $s = 0 \Rightarrow 1(\hat{p}(x)) = 1 \Rightarrow \hat{y}$ predicts all positive

Bagging

1. We fit a lot of high variance low bias models from bootstrap samples. Then we average them together to reduce variance.
2. $\hat{f}_b ag(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^b(x)$
3. Typically choose B large enough for the procedure to settle down.
4. We don't need to do CV. We will automatically have B/3 out of bag observations.

Random Forests

1. Bagging trees are correlated because data are drawn from the same sample.
2. RF does not consider all P available predictors for each tree. This decorrelates trees.
3. In each bootstrap draw, use only m ¡ P predictors. $m = \sqrt{P}$ for classification and $m = P/3$ for regression problems.
4. Bagging is RF when m = P

Boosting

1. Fit lots of small, low variance high bias models, aggregate them slowly to improve forecasting
2. Basis function expansions take the form $f(x) = \sum_{m=1}^{M} \beta_m b(\boldsymbol{x}; \gamma_m)$, $b(x;\gamma)$ are simple functions, γ are parameters in that simple function
3. These models are fit by minimizing a loss function averaged over the training data
$min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1} + \beta b(x_i;\gamma))$

FSAM

1. Initialize $f_0(x) = 0$
2. For m = 1 to M:
a. compute: $argmin_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1} + \beta b(x_i;\gamma))$
b. set $f_m(x) = f_{m-1}(x) + \beta_m b(x, \gamma_m)$
3. The final model is just
$f(x) = f_M(x) = \sum_{m=1}^{M} \beta_m b(x, \gamma_m)$

Gradient Boosting

1. Initialize $f_0(x) = argmin_{\gamma} \sum L(y_i, \gamma)$ for a suitable loss function
2. For m = 1 to M:
a. compute $r_{im} = -\frac{\partial L(y, f(x))}{\partial f(x)}$
b. fit a base learner $h_m(x)$ to r_{im}
c. compute coefficient for $h_m(x)$ as:
$\gamma_m = argmin_{\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i))$
d. update model to
$f_m(x) = f_{m-1}(x) + \lambda \gamma_m h_m(x)$ for $\lambda \leq 1$
3. The final model is just $f_M(x)$

Boosting Regression Trees

1. Initialize $f_0(x) = 0$
2. For m = 1 to M:
a. compute $r_{im} = y_i - f_{m-1}(x_i)$
b. fit a regression tree to residuals
c. for each node, compute
$\gamma_{jm} = argmin_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$
$\gamma_{jm} = \frac{\sum r_{im}}{\text{Number of observations in } R_{jm}}$
d. update:
$f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J_m} \gamma_{jm} 1(x \in R_{jm})$
3. The prediction model is then $f_M(x)$

Loss for K-class Classification

1. Multinomial logit:
$Pr(Y = k|X = x) = \frac{exp(f_k(x))}{\sum_{j=1}^{K} exp(f_j(x))}$
2. Multinomial deviance:
$L(y, p(x)) = -\sum_{k=1}^{K} 1(y = k) f_k(x) + log(\sum_{j=1}^{K} exp(f_j(x)))$

Gradient Boosting for K-class problems

1. Initialize $f_{k0}(x) = 0$
2. For m = 1 to M:
a. set $p_k(x) = \frac{exp(f_{k,m-1}(x))}{\sum_{j=1}^{K} exp(f_{j,m-1}(x))}$
b. for k = 1,..., K
i. compute $r_{ikm} = y_{ik} - p_k(x_i)$
ii. fit a tree to r_{ikm}
iii. $\gamma_{jkm} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{jkm}} r_{ikm}}{\sum_{x_i \in R_{jkm}} |r_{ikm}|(1-|r_{ikm}|)}$
iv. update
$f_{km}(x) = f_{k,m-1}(x) + \lambda \sum_{j=1}^{J_m} \gamma_{jkm} 1(x \in R_{jkm})$
3. Output $\hat{f}_k(x) = f_{kM}(x)$

Relative Importance

1. $\mathcal{I}_l^2(T) = \sum_{t=1}^{J-1} {}^2_t 1(v(t) = l)$
This is the importance of X_l in a tree.
2. $\mathcal{I}_l^2 = \sum_{m=1}^{M} \mathcal{I}_l^2(T_m)$
This is the importance of X_l in a forest.
3. Average importance across trees in a forest:
$\mathcal{I}_{\mathbb{K}}^2 = \frac{1}{M}\sum_{m=1}^{M} \mathcal{I}_{lk}^2(T_{km})$
4. Average importance across forests:
$\mathcal{I}_l^2 = \frac{1}{K}\sum_{k=1}^{K} \mathcal{I}_{\mathbb{K}}^2$

ANN

1. The central idea is to extract linear combinations of the inputs as derived features, and then model the outcome as a nonlinear target of these features.
2. NN is a fancy nonlinear regression model which can be neatly represented by network diagrams.
$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X)$, linear comb. of X into activation function
$T_k = \beta_{0k} + \beta_k^T Z$, linear comb. of Z
$f_k(X) = g_k(T)$, output function from T

Activation Functions

1. $\sigma(v) = 1/(1 + e^{-v})$
2. $\sigma(z) = max\{0, z\}$

Output Functions

1. For regression, usually is the identity function:
$g_k(T) = T_k$
2. For classification, usually the softmax function:
$g_k(T) = \frac{e^{T_k}}{\sum_{l=1}^{K} e^{T_l}}$

Fitting NN

1. For regression, the measure of fit is the squared loss:
$R(\theta) = \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2$
2. For classification, the measure of fit is deviance:
$R(\theta) = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik} log f_k(x_i)$
Let $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$
$R(\theta) = \sum_{i=1}^{N}\sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2$
$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{mi} = \delta_{ki} z_{mi}$
$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il} = s_{mi} x_{il}$
$\beta^{(}_{km} r + 1) = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}}$
$\alpha^{(}_{ml} r + 1) = \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}}$
$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki}$
1. Forward pass: the current weights are fixed and the predictions $\hat{f}_k(x_i)$ are computed from the network.
2. Backward pass: the errors δ_{ki} are computed, and then back-propagated via the equation above to get the errors s_{mi}.
3. Then, both sets of errors are used to compute the gradients for the updating steps.

Dealing with Overfit

1. Early stopping: stop before the global minimum.
2. Regularization: $R(\theta) + \lambda J(\theta)$, where $J(\theta) = \sum_{k,m} \beta_{km}^2 + \sum_{m,l} \alpha_{ml}^2$

Input Scaling

1. ANN is sensitive to the scaling of the inputs
2. It is best to normalize all predictors

Hybrid Learning

1. Firstly, fit LASSO and produce $\hat{f}(x)_{lasso}$
2. Secondly, compute the residuals
$r_i = Y_i - \hat{f}(x_i)_{lasso}$ and train the random forest on this residuals using the same predictors.
3. Form the hybrid prediction:
$\hat{f}(x) = \hat{f}(x)_{lasso} + \hat{f}(x)_{rf}$

Ensemble Learning

$f(x) = \sum_{k=1}^{K} w_k f_k(x)$
$f = w f_1 + (1 - w) f_2$
$Var(f) = w^2 \sigma_1^2 + (1 - w)^2 \sigma_2^2$
$w^* = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} = \frac{\sigma_1^{-2}}{\sigma_1^{-2} + \sigma_2^{-2}}$
$Y_i = \beta_1 f_{1i} + \beta_2 f_{2i} + \ldots + \beta_K f_{Ki} + e_i$
$Min(RSS + \lambda \sum_{i=1}^{K} |\beta_i|)$