Resampling

Resampling Methods

- Repeatedly drawing samples from a training set and refitting a model on each sample to obtain more information about the fitted model.

To avoid biased selection of datapoints from a sample, it may be possible that the data is 'imbalanced'

- Example: Estimate the variability of a linear regression fit by repeatedly drawing different samples from the training data, fitting a regression model to each new sample, and then examining the extent to which the resulting fits differ.

Resampling Methods

- Model Assessment: having chosen a final model, estimating its prediction error on new data.

a) Model Assessment:

Suppose, we have chosen linear regression. We test the error rates, & accuracy, etc for diff. samples of test data.

e.g.: Test data:

$\{1, 2, 3, 4, 5, 6\}$

Taking diff. subset of test data, would give diff. error rate.

b) Model Selection:-

Selecting different models to choose the best one.

Model Assessment (cont.)

Test Error

The average prediction error of a machine learning method on new observations.
The prediction error over an independent test sample.

Training Error

The average loss over the training sample:

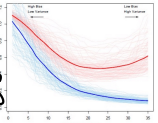
$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}(x_i))$$

Note: The training error rate can dramatically *underestimate* the test error rate

- A model is preferred if it gives better results (more accurate) with new (test) data.

Model Assessment (cont.)

∴ Training error is not good for assessing model.



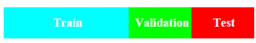
becomes more flexible

- As the model becomes more and more complex, it uses the training data more and is able to adapt to more complicated underlying structures.
- Hence, there is a decrease in bias but an increase in variance. (overfitting)
- However, training error is not a good estimate of the test error.

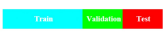
- Training error consistently decreases with model complexity. (complexity ↑ flexibility ↑)
- A model with zero training error is overfit to the training data and will typically generalize poorly.

Model Assessment (cont.)

- If we are in a data-rich situation, the best approach for both *model selection* and *model assessment* is to randomly divide the dataset into three parts: training set, validation set, and test set.



- The *training set* is used to fit the models. The *validation set* is used to estimate prediction error for model selection. The *test set* is used for assessment of the prediction error of the final chosen model.



- A typical split might by 50% for training, and 25% each for validation and testing.

- For example, in case of a regression (prediction) problem, we use 50% of the data to fit and train diff. models like Lasso Reg., LSTM, ARIMA. → train set

- Then we test (calculate accuracy) of each model for validation set.

- At last we use the test set for predicting values using model which gave best accuracy for validation set.

Model Assessment (cont.)

Here, we consider cross-validation (CV) methods that estimate the test error by *holding out* a subset of the training observations from the fitting process, and then applying the machine learning method to those held out observations.

- The most important use of validation is for *model selection*
- In almost every learning situation, there are some choices to be made and we need a principled way of making these choices. { for ex: the best K to be chosen, the best split }
- The leap is to realize that *validation* can be used to estimate the out-of-sample error for more than one model.

(1) The training data trains the model.

(2) The validation data is an integral part as it checks how well the model is trained

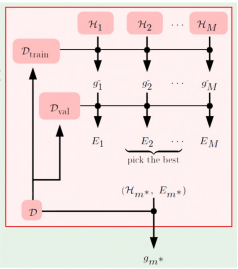
(3) After this stage, we have a better model, which is again used on some unseen (test) data.

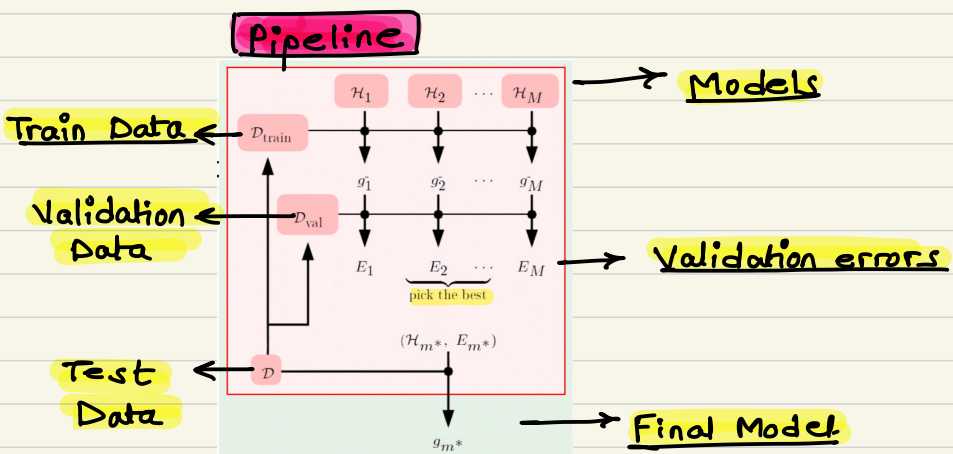
Overview: Model Selection (cont.)

Suppose we have M models; validation can be used to select one of these models.

We use the training data to fit the model, and we evaluate each model on the validation set to obtain the validation errors.

It is now a simple matter to select the model with the lowest validation error.



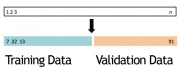


Validation Set Approach

We wish to find a set of variables that give the lowest *validation error rate* (an estimate of the *test error rate*).

If we have a large data set, we can randomly split the data into separate training and validation data sets.

Then, we use the training data set to build each possible model and select the model that gives the lowest error rate when applied to the validation data set.



Validation Set Approach: Example

Example: we want to predict *mpg* from *horsepower*

Two models:

- mpg ~ horsepower
- mpg ~ horsepower + horspower^2

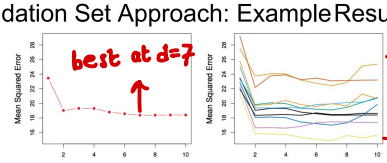
Which model gives a better fit?

We randomly split (50/50) *392 observations* into training and validation data sets, and we fit both models using the training data.

Next, we evaluate both models using the validation data set.

Winner = model with the *lowest* validation (testing) MSE

Validation Set Approach: Example Results



Using single split (say 50-50)

Different MSE for diff. split bet^n train and validation.

Not every split gives min. MSE at d=7

Note:

① In above example, degree = 2 and 3.

② The above plots represent validation MSE for different degree of eq^n.

Validation Set Approach: Review

Advantages:

- Conceptually simple and easy implementation.

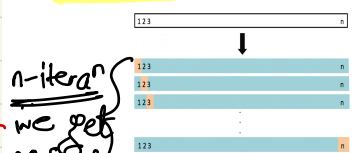
Drawbacks:

- The validation set error rate (MSE) can be highly variable.
- Only a subset of the observations (those in the training set) are used to fit the model.
- Machine learning methods tend to perform worse when trained on fewer observations.
- Thus, the validation set error rate may tend to *overestimate* the test error rate for the model fit on the entire data set.

To overcome this problem we use:

Leave-One-Out Cross-Validation

Instead of two subsets of comparable size, a single observation is used for the validation set and the remaining observations ($n - 1$) make up the training set.



Taking average of n-MSEs gives the MSE of the model.

- We can shuffle the data ONCE BEFORE starting the iterations

Leave-One-Out Cross-Validation

(LOOCV)

- **LOOCV Algorithm:**
  - Split the entire data set of size *n* into:
    - Blue = training data set
    - Beige = validation data set
  - Fit the model using the training data set
  - Evaluate the model using validation set and compute the corresponding MSE.
  - Repeat this process *n* times, producing *n* squared errors. The average of these *n* squared errors estimates the test MSE.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i$$

MSE_i

Validation Set Approach vs. LOOCV

LOOCV has far less bias and, therefore, tends not to overestimate the test error rate.

Performing LOOCV multiple times always yields the same results because there is no randomness in the training/validation set splits.

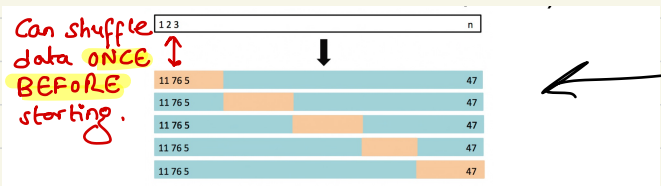
LOOCV is computationally intensive because the model has to be fit *n* times.

Not affordable with large value of 'n'.

K-Fold Cross Validation

- To avoid time complexity of LOOCV we use K-Fold CV.

- Here we choose K windows (folds) as validation set.

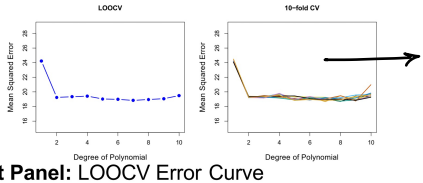


- Takes less time than LOOCV.

- LOOCV is special case of K-fold. ($K = n$)

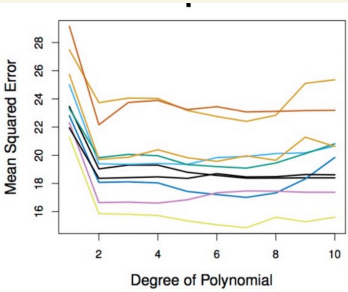
- The first fold is treated as a validation set, and the method is fit on the remaining $K - 1$ folds. → train set The MSE is computed on the observations in the *held-out* fold. The process is repeated K times, taking out a different part each time.
- By averaging the K estimates of the test error, we get an estimated validation (test) error rate for new observations.

K-Fold Cross-Validation vs. LOOCV



Left Panel: LOOCV Error Curve
Right Panel: *10*-fold CV run nine separate times, each with a different random split of the data into ten parts.
Note: LOOCV is a special case of *K*-fold, where $K = n$

It can be observed that the errors reduce upon cross validation



Bias-Variance Trade-off for *K*-Fold Cross-Validation

Which is better, LOOCV or *K*-fold CV?

- However, LOOCV has higher variance than *K*-fold CV (when $K < n$):
- It doesn't *shake up* the data enough. The estimates from each fold are highly correlated and hence their average can have high variance.
- Thus, we see the bias-variance trade-off between the two resampling methods

Bias-Variance Trade-off for *K*-Fold Cross-Validation

K-fold CV with $K = 5$ or $K = 10$ are commonly used, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance

Cross-Validation on Classification Problems

LOOCV Error Rate:

$$\mathrm{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Err_i \text{, where } Err_i = I(Y_i \neq \hat{Y}_i)$$

We use CV as follows:

- Divide data into *K* folds; hold-out one part and fit using the rest (compute error rate on hold-out data); repeat *K* times.
- CV Error Rate: average over the *K* errors we have computed

$$I(Y_i \neq \hat{Y}_i) = 1 \quad (Y_i \neq \hat{Y}_i)$$
$$= 0 \quad (Y_i = \hat{Y}_i)$$

If data is imbalanced, i.e. predicts more false positive/negatives then above measure would not be useful.

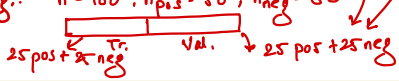
Cross-Validation on Classification Problems: Class Imbalance

Note1: prediction accuracy is not always a good measure of how well a classifier performs. Hence, for highly imbalanced data sets, one can estimate other measures such as precision, recall, or F scores, using cross validation

Cross-Validation on Classification Problems: Class Imbalance

Note 2: when data is highly imbalanced, some of the folds may contain few points or even no points from a rare class. Therefore, sometimes *stratified cross validation* is used, which seeks to ensure that each fold has the same ratio of each class as the original data set.

eg.: $n = 100$; $n_{pos} = 50$; $n_{neg} = 50$



25 pos + 25 neg → 25 pos + 25 neg

Cross-validation: right and wrong

- Consider a simple classifier applied to some two-class data:
  1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels.
  2. We then apply a classifier such as logistic regression, using only these 100 predictors.

How do we estimate the test set performance of this classifier?

Can we apply cross-validation in step 2, forgetting about step 1?

Ans: NO!

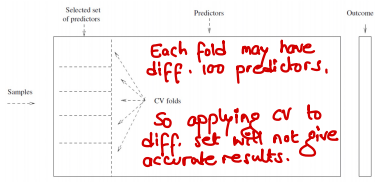
Applying cross-validation in step 2, we may get different set of 100 predictors which were not a part of train set

- This would ignore the fact that in Step 1, the procedure *has already seen the labels of the training data*, and made use of them. This is a form of training and must be included in the validation process.

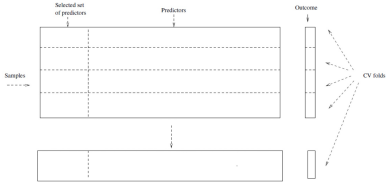
The Wrong and Right Way

- *Wrong:* Apply cross-validation in step 2.
- *Right:* Apply cross-validation to steps 1 and 2.

Cross-Validation: Wrong Way



Cross-Validation: Right Way



https://carpentries-incubator.github.io/machine-learning-novice-python/07-bootstrapping/index.html

* Bootstrap.

The Bootstrap

- The *bootstrap* is used to quantify uncertainty associated with a given estimator or machine learning method; it is a general tool for assessing statistical accuracy.

① Logistic Reg.
② KNN Classif.
③ . . . .

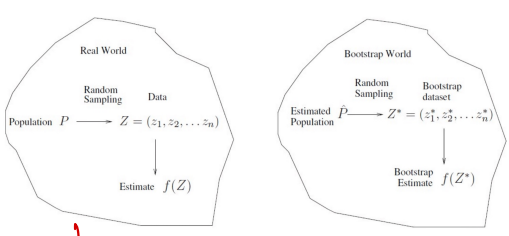
- It is a resampling method and involves repeated drawing of samples (w/ replacement) to estimate a parameter (eg. min, mean, std.)

The Bootstrap

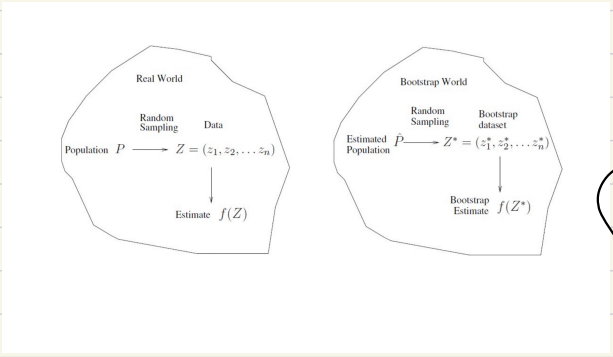
- As an example, it can be used to estimate the standard errors of the coefficients from a linear regression fit, or a confidence interval for that coefficient.
- The use of the term bootstrap derives from the phrase "to pull oneself up by one's bootstraps."

→ ★ Confidence interval:

In frequentist statistics, a **confidence interval** (**CI**) is a range of estimates for an unknown parameter. A confidence interval is computed at a designated *confidence level*; the 95% confidence level is most common, but other levels, such as 90% or 99%, are sometimes used.[1][2] The **confidence level** represents the long-run proportion of corresponding CIs that contain the true value of the parameter. For example, out of all intervals computed at the 95% level, 95% of them should contain the parameter's true value.[3]



Monte-Carlo simulation



Bootstrapping is Monte-Carlo simulation over histograms of training data

Histogram ~ Empirical Distribution

The Bootstrap: Overview (cont.)

Suppose we have a model fit to a set of training data. We denote the training set by $\mathbf{Z} = (z_1, z_2, \ldots, z_N)$ where $z_i = (x_i, y_i)$.

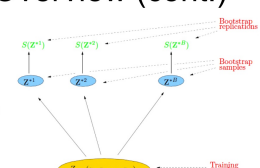
The basic idea is to randomly draw datasets **with replacement** from the training data, each sample the same size as the original training set.

This is done B times, producing B bootstrap datasets. Then we refit the model to each of the bootstrap datasets, and examine the behavior of the fits over the B replications.

The Bootstrap: Overview (cont.)

$S(\mathbf{Z})$ is any quantity computed from the data $\mathbf{Z}$, for example, the prediction at some input point.

From the bootstrap sampling we can estimate any aspect of the distribution of $S(\mathbf{Z})$, for example, its variance:



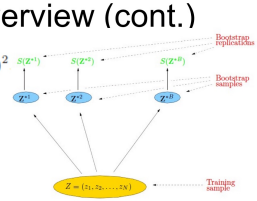
$$\widehat{\mathrm{Var}}[S(\mathbf{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\mathbf{Z}^{*b}) - \bar{S}^*)^2$$

$$\bar{S}^* = \textstyle\sum_b S(\mathbf{Z}^{*b})/B$$

The Bootstrap: Overview (cont.)

$$\widehat{\mathrm{Var}}[S(\mathbf{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\mathbf{Z}^{*b}) - \bar{S}^*)^2$$

$$\bar{S}^* = \textstyle\sum_b S(\mathbf{Z}^{*b})/B$$



Note that this estimated variance can be thought of as a Monte-Carlo estimate of the variance of $S(\mathbf{Z})$ under sampling from the empirical distribution function.

The Bootstrap: An Example

Suppose that we wish to invest a fixed sum of money in two financial assets that yield returns X and Y; note that X and Y are random quantities.

We will invest a fraction α of our money in X, and will invest the remaining $1 - \alpha$ in Y.

Goal: Since there is variability associated with the returns on these two assets, we wish to choose α to minimize the total risk, or variance, of our investment.

1

The Bootstrap: An Example (cont.)

In other words, we wish to minimize $\mathrm{Var}(\alpha X + (1 - \alpha)Y)$=

The value that minimizes the risk, in this examples, is:

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}$$

where

$$\sigma_X^2 = \mathrm{Var}(X), \sigma_Y^2 = \mathrm{Var}(Y), \text{ and } \sigma_{XY} = \mathrm{Cov}(X, Y)$$

2

The Bootstrap: An Example (cont.)

In reality, the quantities σ_X^2, σ_Y^2, and σ_{XY} are unknown, so we can compute estimates for these quantities using a data set that contains past measurements for X and Y. We can then estimate the value of α that minimizes risk using:

3

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}$$

The Bootstrap: An Example (cont.)

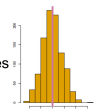
To do so, the process of simulating 100 paired observations of X and Y is repeated 1,000 times; now we have 1,000 estimates for α.

The mean over all 1,000 estimates for α →

$$\bar{\alpha} = \frac{1}{1000}\sum_{r=1}^{1000} \hat{\alpha}_r = 0.5096$$

The standard deviation of the estimates →

$$\sqrt{\frac{1}{1000-1}\sum_{r=1}^{1000}(\hat{\alpha}_r - \bar{\alpha})^2} = 0.083$$

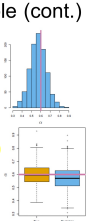


4

The Bootstrap: An Example (cont.)

Each of these *bootstrap data sets* is created by sampling *with replacement*, and is the *same size* as the original data set.

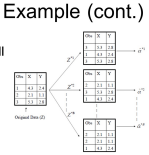
As a result, some observations may appear more than once in a given bootstrap data set and some not at all.



1

The Bootstrap: An Example (cont.)

A graphical illustration of the bootstrap approach on a small sample containing $n = 3$ observations.



Each bootstrap data set contains n observations, sampled **with replacement** from the original data set.

Each bootstrap data set is used to obtain an estimate of α.

2

The Bootstrap: An Example (cont.)

Denoting the first bootstrap data set by Z^{*1}, we can use Z^{*1} to produce a new bootstrap estimate for α, which we call $\hat{\alpha}^{*1}$

This procedure is repeated B times for some large value of B (say 100 or 1000), in order to produce B different bootstrap data sets, Z^{*1}, Z^{*2},…, Z^{*B}, and B corresponding α estimates, $\hat{\alpha}^{*1}$,$\hat{\alpha}^{*2}$,...,$\hat{\alpha}^{*B}$.

3

The Bootstrap: An Example (cont.)

We compute the standard error of these bootstrap estimates using:

$$\mathrm{SE}_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1}\sum_{r=1}^{B}(\hat{\alpha}^{*r} - \hat{\alpha}^*)^2}$$

This serves as an estimate of the standard error of $\hat{\alpha}$ estimated from the original data set.

4

Caution :

For example, if the data is a time series, we cannot simply sample the observations with replacement.

→ The sequence of data is important. So cannot use Bootstrap.

However, we can instead create blocks of consecutive observations, and sample those with replacement. Then, we paste together sampled blocks to obtain a bootstrap dataset.

Soln

example :

Bootstrap Confidence Intervals ★★

Easy Algorithm for Building (1-α) Confidence Interval Using Bootstrap

1. Create B bootstrap datasets from original dataset with the same size N.
2. Calculate the mean of each bootstrap sample $m_1, m_2, \ldots, m_B$. Consider them as a sample of the sample means.
3. Order the sample means and call them $m_{(1)}, m_{(2)}, \ldots, m_{(B)}$
4. The middle (1-α)B of the sample yield the (1-α) Confidence interval for the mean

Bootstrap Percentile Confidence Interval: Example

Assume the sample data is
30, 37, 36, 43, 42, 43, 43, 46, 41, 42

Problem: Estimate the mean μ of the underlying distribution and give an 80% bootstrap confidence interval.

Solution: $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

$= 40.3$

Confidence interval can be calculated for any feature (mean, median, std, IR)

Bootstrap Percentile Confidence Interval: Example

20 bootstrap samples were generated, each of size 10. Each of the 20 columns in the following array is one bootstrap sample.

43	36	46	30	43	43	43	37	42	42	43	37	36	42	43	43	42	43	43	43
43	41	37	37	43	43	46	36	41	43	43	42	41	43	46	36	43	43	43	42
42	43	37	43	46	37	36	41	36	43	41	36	37	30	46	46	42	36	36	43
37	42	43	41	41	42	36	42	42	43	42	43	41	43	36	43	43	41	42	46
42	36	43	43	42	37	42	42	42	46	30	43	36	43	43	43	42	37	36	42	30
36	36	42	42	36	36	43	41	30	42	37	43	41	41	43	43	42	46	43	37
43	37	41	43	41	42	43	46	46	36	43	42	43	30	41	46	43	46	30	43
41	42	30	42	37	43	43	42	43	43	46	43	30	42	30	42	30	43	43	42
46	42	42	43	41	42	30	37	30	42	43	42	43	37	37	37	42	43	43	46
42	43	43	41	42	36	43	30	37	43	42	43	41	36	37	41	43	42	43	43

μ_1 μ_2 μ_3 ... μ_{20}

38.7 38.9 38.9 39.4 39.8 40.1 40.2 40.4 40.5 40.5 40.7 40.7 41 41.2 41.4 41.5 41.5 41.8 41.9 42.3

Middle 80% values ⇒ $[\mu_3, \mu_{18}]$

∴ Mean will lie betⁿ [38.9, 41.9] → Confidence Interval.

The Bootstrap: More Details (cont.)

- Note that about 2/3 of the original N data points appear in each bootstrap sample:

$$\left\{ \begin{array}{l} \Pr\{\text{Observation } i \in \text{bootstrap sample } b\} \\ = 1 - \Pr\{\text{Observation } i \notin \text{bootstrap sample } b\} \end{array} \right\}$$

- One the other hand

$$\left\{ \begin{array}{l} \Pr\{\text{Observation } i \notin \text{bootstrap sample } b\} \\ = \Pr\{\text{Observation } i \text{ is not any of the } N \text{ members of the bootstrap sample } b\} \end{array} \right\}$$

Pr{obs i ∉ bootstrap sample b} $= (1 - 1/N)^N = e^{-1}$

★ ∴ Pr(Obs. i ∈ bootstrap b) $\approx 1 - e^{-1} \approx 0.632$

The Bootstrap: More Details (cont.)

This overlap can make overfit predictions like unrealistically good, and is the reason that cross-validation explicitly uses non-overlapping data for the training and test samples.

In other words, this will cause the bootstrap to seriously *underestimate* the true prediction error.

By mimicking cross-validation, a better bootstrap estimate can be obtained.

The Bootstrap: More Details (cont.)

For each observation, we only keep track of prediction from bootstrap samples not containing that observation.
The leave-one-out bootstrap estimate of prediction error is defined by:

$$\widehat{\mathrm{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i))$$

Here C^{-i} is the set of indices of the bootstrap samples b that do *not* contain observation i, and $|C^{-i}|$ is the number of such samples.

a) $$\Pr(Y = \text{orange} \mid X = x) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 x}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 x}}$$

$$\log \text{odds} : \log\left(\frac{p(x)}{1 - p(x)}\right)$$

$$= \boxed{\hat{\beta}_0 + \hat{\beta}_1 x}$$

b) $$\Pr(Y = \text{Orange} \mid X = x) = \frac{e^{(\hat{\alpha}_{\text{orange}0} + \hat{\alpha}_{\text{orange}1} x)}}{e^{(\hat{\alpha}_{\text{orange}0} + \hat{\alpha}_{\text{orange}1} x)} + e^{(\hat{\alpha}_{\text{apple}0} + \hat{\alpha}_{\text{apple}1} x)}}$$

Let $\hat{\alpha}_{\text{orange}0} = a$, $\hat{\alpha}_{\text{orange}1} = b$

$\hat{\alpha}_{\text{apple}0} = c$, $\hat{\alpha}_{\text{apple}1} = d$

$$\Rightarrow \Pr(Y = \text{orange} \mid X = x) = \frac{e^{a+bx}}{e^{a+bx} + e^{c+dx}}$$

$$\Rightarrow \Pr(Y = \text{apple} \mid X = x) = 1 - \frac{e^{a+bx}}{e^{a+bx} + e^{c+dx}}$$

$$= \frac{e^{c+dx}}{e^{a+bx} + e^{c+dx}}$$

$$\therefore \log \text{odds} = \log\left(\frac{\Pr(\text{orange})}{\Pr(\text{apple})}\right)$$

$$= (a + bx) - (c + dx)$$

c) We cannot comment on this, as we don't know the values of :

$$\hat{\alpha}_{orange,0}, \hat{\alpha}_{orange,1}, \hat{\alpha}_{apple,0}, \hat{\alpha}_{apple,1}$$

d) $\hat{\beta}_0 = \hat{\alpha}_{orange,0} - \hat{\alpha}_{apple,0}$

$= 1.2 - 3$

$= -2.2$

$\hat{\beta}_1 = \hat{\alpha}_{orange,1} - \hat{\alpha}_{apple,1}$

$= -2 - 0.6$

$= -2.6$

e) Softmax Coding is similar to normal logistic regression in every aspect, so predicted class labels for both mine and my friends models will always be same.

3. Consider a logistic regression problem in which there are no features, which means that:

$$\Pr(Y=1) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$$

Assume that we have m data points with label $Y=1$ and n data points with label $Y=0$ (remember that features are irrelevant).

(a) Write down the likelihood function $l(\beta_0)$.

(b) Find the Maximum Likelihood estimate $\hat{\beta}_0$ for this data set. [**Hint**: maximize $\log_e l(\beta_0)$].

(c) Determine conditions under which this simple classifier classifies data points into $Y=1$ or $Y=0$.

a)

$$l(\beta_0) = \prod_{i:y_i=1} P(Y=1|X=x) \prod_{i:y_i=0} P(Y=0|X=x)$$

$$= \frac{m}{m+n}\left(\frac{e^{\beta_0}}{1+e^{\beta_0}}\right) \cdot \frac{n}{m+n}\left(\frac{1}{1+e^{\beta_0}}\right)$$

$$= \frac{mn}{(m+n)^2} \frac{e^{\beta_0}}{(1+e^{\beta_0})^2}$$

b) Maximum Likelihood :

$$\log_e l(\beta_0) = \log\left(\frac{mn}{(m+n)^2} \frac{e^{\beta_0}}{(1+e^{\beta_0})^2}\right)$$

$$= \log(mn) - 2\log(m+n) + \beta_0 - 2\log(1+e^{\beta_0})$$

c)

$$P(Y=1) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$$

$\beta_0 \to -\infty \Rightarrow P(Y=1) \to 0$

$\beta_0 \to +\infty \Rightarrow P(Y=1) \to 1$

$\therefore$ As $\beta_0 \to -\infty$ it will point to class 0

and As $\beta_0 \to +\infty$ it will point to class 1

☆ Difference bet^n KNN classifier and KNN regression.

Ans. :

① Codomain (output space) of KNN reg. is continuous. $Y \in \mathbb{R}$

whereas, for KNN reg., codomain is discrete and $Y \in [0, 1]$

② For KNN reg. o/p variable is quantitative
For KNN class. o/p variable is qualitative (categorical)

③

$$\hat{y}_i = x_i \hat{\beta}$$

$$\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$$