Machine Learning Exam prep

1. Supervised learning → data Example with label is feeded
2. Unsupervised learning → data example without label is feeded.

→ $Y = f(X) + \varepsilon$

Y → output label

X → feature

f → model लाई train गर्नु भनेको, f का all parameters हरू calculate गर्नु हो।

→ loss is penalty for bad prediction. जुन model को loss minimum हुन्छ त्यो best model हुन्छ।

Basic assumption of ML models

1. Examples are drawn iid from the distribution
2. distribution is stationary, i.e distribution does not change within the data set.
3. we draw example from partition from the same distribution.

Loss for regression setting

$$MSE = \frac{1}{n} \sum (y_i - \hat{f}(x_i))^2 \quad \text{(Numerical)}$$

Loss for classification setting

$$\text{Error rate} = \frac{1}{n} \sum I(y_i \neq \hat{f}(x_i))$$

$y_i \neq \hat{f}(x_i)$ → statement true ⇒ output = 1

→ Model complexity increase भयो भने, training error decrease and decrease but test error first decrease and then increase.

→ Our goal is to select a mode low variance and low bias as far as possible.

Variance

→ यदी हामीले हाम्रो training data change गर्नौं हो भने, prediction कसरी change हुदैछ

Bias

→ wrong prediction on our data (or missclassification) because of using too simple model.

KNN-Classifier

① Good classifier भनेको त्यो हो जसको लागी ~~error~~ test error rate minimum हुनछ

② The lowest error rate we can get is given by a very simple classifier called Bayes classifier. (alway greater then zero, it is kind of irreducible error)

L_p Norm → Minkowski distance
L_2 Norm → Euclidean distance
L_1 Norm → Manhatten distance.
} for numerical variable.

Hamming distance → for categorical Variable

Grower distance → for both Numerical & categorical.

→ When one feature value are large then other that feature will dominate the distance & hence the outcome of KNN.

→ data मा outlier छ भने we prefer standardization rather than normalization.

→ Small K value : Low bias but high variance

→ Large K value : low variance but high Bias.

→ KNN is non-parametric (does not care about distribution of data)

Logistic Regression.

→ यदी predict गर्ने variable numerical छ भने, no problem we will use linear regression, but यदी predict गर्ने variable categorical छ भने use Logistic regression.

$$P(X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\cdots+\beta_p X_p)}}$$

P(class/feature)

$P(Y=k \mid X=x)$

→ will never give exactly zero and exactly one.

$$\underbrace{\log\left[\frac{P(X)}{1-P(X)}\right]}_{\text{Log-odds or logit.}} = \beta_0+\beta_1 X_1+\cdots+\beta_p X_p$$

→ logistic regression is linear regression with target log-odds.

→ logistic model specify गर्दा family = binomial लेख्नु पर्छ।

→ logistic model categorical variable को लागि use

→ यदी only two class छ भने categorical logistic model but if we have more than two classes for categorical variable ⇒ use discriminant Analysis.

Linear discriminant Analysis (LDA):

→ LDA is good when there is more than two class for categorical variable. P(feature/class) i.e P(X/Y)

→ जब classes are well seperated, this works nicely.

→ यदी sample size (data size) n is small and normally distributed हो जाते, it works more nicer

Suppose $Y \longrightarrow$ has K classes

→ $\pi_K = P(Y=K)$ is called prior probability. This is the probability that randomly choosen observation comes from the K^{th} class.

$P(Y=K/X=x)$ is called posterior probability. This is the probability that an observation belongs to the K^{th} class, given the feature value for that observation. (Here we do not directly compute this)

→ Each ~~calls~~ class has its own mean (μ_K) but common variance σ^2 (In case of LDA)

→ $\sum_{i=1}^{K} \pi_K = 1$

Linear in X (so will have linear seperator)

$$\hat{\delta}_K(X) = X \cdot \frac{\hat{\mu}_K}{\hat{\sigma}^2} - \frac{\hat{\mu}_K^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_K)$$ } discriminant function

→ assign an observation to class 1 if $2X(\hat{\mu}_1^2 - \hat{\mu}_2^2) > \hat{\mu}_1 - \hat{\mu}_2$

i.e $X > \dfrac{\hat{\mu}_1 + \hat{\mu}_2}{2}$

→ so $X = \dfrac{\hat{\mu}_1 + \hat{\mu}_2}{2}$ is the decision boundary.

→ for multi-dimensional feature data $\sim N(\mu_K, \Sigma)$

→ $\hat{\delta}_K(X) = \overset{\text{linear}}{X^T} \hat{\Sigma}^{-1}\hat{\mu}_K - \frac{1}{2}\hat{\mu}_K^T \hat{\Sigma}^{-1}\hat{\mu}_K + \log(\hat{\pi}_K)$

Quadratic Discriminant Analysis

→ ऊपर assumption are same like LDA i.e the observations from each class are drawn from a normal distribution. But here each class will have different mean and different covariance matrix also.

→ data $\sim N(\mu_K, \Sigma_K)$

→ $\hat{\delta}_K(X) = -\frac{1}{2} X^T \hat{\Sigma}_K^{-1} X + X^T \hat{\Sigma}_K^{-1}\hat{\mu}_K - \frac{1}{2}\hat{\mu}_K^T \hat{\Sigma}_K^{-1}\hat{\mu}_K + \log(\hat{\pi}_K)$

quardatic in X.

Naive Bayes

→ unlike LDA and QDA classifier that assumes the probability distribution $P(X/\text{class})$ follows a p-dimensional multivariate normal distribution, Naive Bayes classifier assumes the p features are independent i.e

$$f_K(X) = f_{K1}(X_1)\, f_{K2}(X_2) \ldots f_{Kp}(X_p)$$

comparison of LDA and Logistic Regression

→ for binary classes both logistic and LDA are closely connected.

→ If your data has gaussian distribution then LDA will do better than Logistic.

→ generally for multiclass logistic is not good, LDA is better.

#= comparision of LDA with KNN

→ KNN is compeletely different, its decision bandry can have any form and does not depends upon distribution of data.

→ KNN will perform best when decision bandry is highly non-linear.

→ KNN does require a larger training set so form an accurate decision boundary.

→ KNN has less interpretability.

Compared with QDA

→ QDA is inbetween LDA and KNN

→ QDA has upto quadratic boundary not limited to linear.

→ QDA performs better then KNN when training set is small. Because it makes assumption of the form of the decision bandary.

→ KNN लाई large data set चाहिन्छ राम्रो संग perform गर्न।

→ LDA ले total error rate लाई कम गर्ने काम गर्दछ, for that it does not care if individual class मा error rate high हुन पनि।

Threshholding

→ किनकी many classification model return probability, you need to choose threashhold to classify them into two groups.

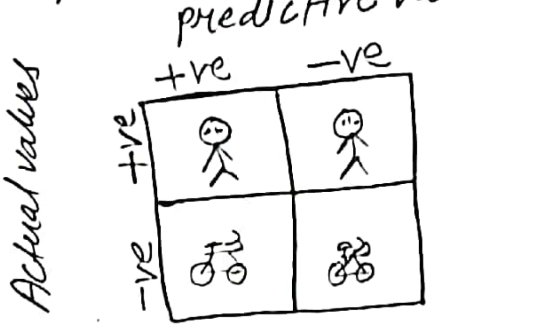
→ Choosing a threshold is assessing how much you will suffer for making a mistake.

Accuracy

$$\text{Accuracy} = \frac{\text{Number of correct prediction}}{\text{Total number of prediction}}$$

→ यदि different kind of mistakes have different costs related to them छ भने, accuracy is poor or misleading metric. (4)
Eg:- for Class Imbalance data.

Confusion Matrix



$$\text{precision} = \frac{TP}{TP+FP}$$ precision is the proportion of positive identifications was actually correct

$$\text{Recall} = \frac{TP}{TP+FN}$$ Recall is the proportion of actual positive was identified correctly.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

ROC curve

→ First कुरा त यो तब मात्र बनाउन पाइन्छ when your classification model gives you a probabilities value.

→ This is actually the graph of performance at different thresholds value

→ it is TPR vs FPR plot where

$$TPR = \frac{TP}{TP+FN} \quad \text{(recall)}$$

&

$$FPR = \frac{FP}{FP+TN}$$

AVC

→ AVC provides an aggregates measure of performance across all possible classification thresholds

→ यो भनेको area under ROC curve हो

→ AVC value close to 1 means your model performs better

Cross-validation

→ Cross-validation is used to estimate the test error associated with a given model. by repetedly sampling from the training set and refitting the model.

→ यो कुनै पनि predictive modeling मा use गर्न सकिन्छ

→ it is used for model assesment i.e evaluating model's performance

→ model parameter tuning मा पनि use गरिन्छ ।

→ test error rate easily calculate गर्न सकिन्छ If we have a designated test set.

note

→ model कस्तो perform गर्ने हो on test set will also depends upon the split of your orginal data set.

→ किनकि your model only depends upon training data set for fitting model, If your training set is not good representative of orginal population, then model will not be good.

→ There are two types of cross-validation

1. Leave-one-out cross-validation (LOOCV)

→ Leave-one-out cross-validation (LOOCV) involves splitting the set of training samples into two parts repeatedly. A test set should still be held out for final evaluation, but we don't need validation set any more while doing this.

→ At i^{th} iteration : From the training data set i^{th} observation is used ~~for~~ as validation set and remaining as training set.

→ so if tranning set has n observation, our model will be fitted in $n-1$ observation and will predict value for that one i^{th} observation.

→ we will get n test error rate and MSE is average of those.

Advantages and disadvantages of LOOCV

→ we have to fit our model almost n times, where n is no. of observation in training set.

→ Time consuming

→ each MSE_i are highly correlated, because model is trained each time on an almost identical set of observation.

2. K-fold cross validation

→ almost same as LOOCV but ~~here~~ we will put onot only 1 aside for validation, we will put one group of k observation aside for validation and remaining for training the data.

comment on K-Fold cross validation

→ LOOCV is a special case of K-fold CV in with $K=n$.

→ K is no. of group you want to split your data set in

→ # of observation in a group $= \dfrac{nrow(Training\_set)}{K}$

Bootstrap

→ we also say that boostrap is a method that use the sample data to generate an approximate distribution of the estimators, rather then just give a single point estimate

→ Boostrap sampling भनेको method हो that involves drawing of sample data repeated with replacement from a data source to estimate a population parameter.

→ एक sample लिने त्यसको sample mean calculate गर्ने, अर्को sample लिने त्यसको sample mean calculate गर्ने, it will vary, and how it varies is uncertain. त्यसैले सबैलाई त्यस्ता sample mean हरूको distribution चाहिने हो called sampling distribution.

→ sampling distribution of many sample कसरी पत्ता लगाउने त?

→ किनकी real world case मा त हामीलाई एक मात्र sample हुन्छ, एक मात्र sample बाट sampling distribution कसरी पत्ता लगाउने।

→ so here comes the idea of bootstrap → आफ्नो observed एक sample लाई population (pseudo-popⁿ) मानेर त्यसबाट sample लिने repeatedly

comments on Bootstrap

→ we don't make any assumption of the population distribution. so bootstrap is completely non-parametric.

→ यदी orginal sample ले population को राम्रो representative हुन भने, bootstrap sample पनि हुने हैन।

Regularization

→ जति overfit हुन उति training error decrease हुँदै जान पनि test error चाहीं increase हुन लाग्छ for complex model. we can prevent this overfitting by penalizing complex models, a principle called regularization.

→ In simple words, complex model लाई penalize गरेर overfitting हुन दिने गर्छन, this is what we call regularization.

→ structural risk minimization: minimize both loss and complexity of model.

regularization rate.

$$\text{minimize}\left(\text{loss}(\text{Data}|\text{model}) + \lambda * \text{complexity}(\text{model})\right)$$

→ Two types of regularization L_1 and L_2

L_2 Regularization and Ridge Regression

→ Ridge regression is sensitive to the scale of predictors

→ कुनै पनि cofficient को value exectly zero हुने हैन

→ त्यसैले final model मा पनि सबै predictors रहन हुनेछन्।
no feature selection will happen.

L_1 Regularization and the LASSO

→ L_1 Regularization मा चाहिँ coefficient can go to exactly zero

→ The feature for which coefficient is zero will be removed from the model, so it can be used for feature selection also. ◇

Comparing Ridge and LASSO

→ Lasso has similar behaviour to ridge regression, in that as λ increases, the variance decreases and the bias increases

→ Lasso smaller no. of predictor को लागि good. Ridge regression is good when there are many predictors, all with coefficients of roughly equal size.

Note α (alpha = 0 → Ridg regression & alpha = 1 → Lasso)

→ In Ridge λ value increase गरेर coefficient लाई close to zero लैजान सकिन्छ but can not be made exactly zero

Decision Tree :-

→ How do we choose regions $R_1, \dots, R_J$?

⇒ we use recursive binary splitting

→ We first select the predictor X_j and the cutpoint s such that splitting into the region $R_1(j,s) = \{X | X_j < s\}$ and $R_2(j,s) = \{X | X_j \geq s\}$ leads to minimum cost.

→ हामीले cut यस्तो गर्न खोज्छौं कि cut गरेर आउने प्रत्येक region मा cost minimum होस।

→ यदी regression tree हो भने जसलाई minimize गर्नु पर्ने ~~tree~~ cost will be SSE (6)

→ यदी classification tree हो भने, जसलाई minimize गर्नु पर्ने cost are

① Gini Index
② Cross-entropy
} other name is impurates

→ $\hat{P}_{mk}$ भनेको proportion of training observation in the mth region that are from the K^{th} class

Advantages of decision tree and disadvantages

→ easy to visualize
→ Complex tree without pruning is likely to overfit the data, leading to poor test set performance.

→ during tree construction it uses greedy search approach, so they are more expensive to train.

→ They are unstable.

→ Note:- Single decision tree always suffers from high Variance.

→ because a single decision tree has high variance, we use multiple decision tree to make decision. For example Bagging and Random forest.

→ take many training set from the population, with these training set train the models. And final prediction will be average of these models

→ since we only have one training set, use the idea of bootstrap

Bagging

→ Bagging can improve predictions for many regression methods, it is particularly useful for decision trees.

→ On average, each bagged tree only makes use of around two-third of the observations.

→ ओ बाँकी 1/3 (not used for fitting) observation are called out-of-bag (OOB) observations.

→ In simple language कुनै पनि observation 1/3 time training मा पर्दै होइन अनि त्यसलाई training set बाट बनेका B model को predict गर्न पाइयो, we can have B/3 predictions.

→ यो OBB error rate work as test error rate.

→ Bagging does not have good interpretability

→ split गर्ने बेला सबै variable लाई consider गर्ने भयो mtry = # of variable

Random Forest

→ at each split in the tree, RF algorithm only $m = \sqrt{p}$ predictors

→ If you do random forest, bagged tree will not be highly correlated.

→ Random forest मा हामीले गर्ने भनेको de-correlating the trees हो

→ Random forest will typically be helpful when we have a large number of correlated predictors

Boosting

→ The basic idea behind boosting is converting many weak learners to form a single strong learner

→ Weak classifier ले random guess भन्दा थोरै राम्रो गर्दछ।

→ Ensembling method भनेको combining several weak learners to from a strong learner.

Types of Ensemble Methods

① Sequential learners :-

Where different models are generated sequentially and the mistakes of previous models are learned by their successors. प्रत्येक चोटि mislabeled example लाई higher weight दिने (Eg. AdaBoast)

② Parallel learners

Where base models are generated in paralles. eg (Random forest)

Boosting

Recall गरौं bagging मा each tree is built on a bootstrap data set, Independent of the other tree. Boosting works in a similar way, except that the tree are grow previously tree (stumps)

→ First मा boosting algorithm assign equal weight to each data sample. ~~and then~~ it feed the data to the first model, called the base algorithm. The base algorithm makes prediction for each data sample.

→ look the model prediction and increase the weight of sample with a more significant error.

→ Then pass this new weighted data to train next decision tree

adaptive Boosting (AdaBoost)

→ यसको प्रत्येक step मा wrongly classified data point को weigh increase गरेर त्यसलाई correctly classify गराउन खोज्छ।

Advantages and Disadvantages of AdaBoost

→ Easier to use with less need for tweaking parameters.

→ Can be extend beyond binary classification.

→ Boosting is an approach that learns slowly, reducing the potentially overfitting issue.

Disadvantage

→ Adaboost is also extremely sensitive to Noisy data and outliers

→ Adaboost has also been proven to be slow.

support vector Machine

→ SVM मा cost parameter C deals with trade off between bias & variance

→ ~~If C is small, Margin is large, more misclassification are allowed which means higher bias but lower variance~~

→ ~~If C is large, margin is small, almost no misclassification, strict on the classification of the training data, which means lower bias but higher variance.~~

→ SVM ले binary class लाई hyperplane ले seperate गर्छ।

→ non-linear class boundries पनि draw गर्न सक्छ

→ can also works for more then two classes.

→ A test observation is assigned a class depending on which side of the hyperplane it is located.

→ यदी training data लाई perfectly seperate हुन्छ using a hyperplane, then त्यस्ता hyperplanes छन त infinitely many exist गर्छ which can seperate your training data. The idea is to choose one with maximal margin called maximal margin hyperplane.

→ hyperplane देखी training observation सम्मको minimum distance लाई margin भनिन्छ।

→ ती जो margin largest भएको hyperplane लाई maximal margin hyperplane भनिन्छ।

→ Training observations that are on the margin are called support vectors.

→ अरु observation यता-उता गरे they can not affect seperating hyperplane but if support vector more गरे, they affect ~~affe~~ seperating hyperplane.

→ maximal margin hyperplane depends upon support vectors

→ every observation need to be on the correct side of both hyperplane & margin.

→ seperating hyperplane exist गर्दैन भन्नुको मतलब for $M>0$, optimization problem has no solution.

→ डाटालाई perfectly ~~classify~~ separate गर्ने classifier चाहिएको होइन

→ डाटा आफै - आफै केही observation लाई incorrect side of margin or even incorrect side of hyper plane जान दिन्छ so that they perform better in test data.

→ C = c determines the number and severity of the violations to the margin that we will tolerate.

→ $\varepsilon_i = 0$ → observation is in the correct side of margin

→ $\varepsilon_i > 0$ → observation violated the margin

→ $\varepsilon_i > 1$ → observation is on the wrong side of the hyperplane (misclassified).

→ $C = 0$ → there is no budget for violations (no observation are allowed violation)

→ $C > 0$ → there can be no more than C observations misclassified

→ larger C ⇒ wider margin

→ only observation that either lie on the margin or that violate the margin (the support vectors) will affect the hyperplane.

→ When the tuning parameter C is large, then we allow many observations violates the margin (More support vectors). In this case, many observation are involved in determining the hyperplane. We have low variance but high bias

$C \uparrow \rightarrow bias \uparrow \Rightarrow variance \downarrow$

→ C is called the "cost", which controls the trade-off betⁿ the slack variable penalty and the margin.

The Kernel trick:-

$X_1 = (1,2)$ and $X_2 = (3,4)$

$\phi(X_1) = (\overset{1^2}{1}, \overset{1\times2}{2}, \overset{2\times1}{2}, \overset{2^2}{4})$ $\quad \phi(X_2) = (\overset{3^2}{9}, \overset{3\times4}{12}, \overset{4\times3}{12}, \overset{4^2}{16})$

distance betⁿ $\phi(X_1)$ and $\phi(X_2)$

$$\langle \phi(X_1), \phi(X_2) \rangle = \phi(X_1) \cdot \phi(X_2)$$

$$= 1\times9 + 2\times12 + 2\times12 + 4\times16$$

$$= 121$$

The kernel Trick

$$(\langle X_1, X_2 \rangle)^2 = (1\times3 + 2\times4)^2$$

$$= (3+8)^2$$

$$= 11^2$$

$$= 121 \text{ (same answer.)}$$

→ SVM এ non-linear boundary থাকলে তখন Kernel trick use করি

→ Three করব না

```
train = sample(1:nrow(OJ), 800)
test = OJ[-train, ]
```

# of terminal node = 80 = To

$$\text{mean deviance} = 0.692 = \frac{\text{dev}}{\underset{\substack{\downarrow \\ \text{Total no. of} \\ \text{training} \\ \text{obb}}}{800} - \underset{\substack{\uparrow \\ \text{Terminal} \\ \text{node}}}{80}} = \frac{452.9}{720}$$

$$\text{Miss-classification error rate} = 0.15 = \frac{\text{\# of misclassification}}{\text{Total no. of training obb}}$$

↳ training error rate.

$$= \frac{120}{800}$$

→ tree का terminal node हेर्नलाई लागी, we have to look the * one $n=7$ ~~means 7th observa~~

no. deviance

∴ deviance = 0.00

→ decision tree and random forest दुवैमा मेरो train set र तेस्तो मेरो set of indicies चाहिए हो।

mtry = 4 ⇒ Number of variable tried at each split.

ORDINARY NONPARAMETRIC BOOTSTRAP

tune function ले by default 10-fold cross-validation use गर्छ।

Equation of hyper-plane is

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

$$\text{intercept} = \frac{\beta_0}{\beta_2}$$

$$\text{Slope} = \frac{-\beta_1}{\beta_2}$$

margin length = smallest distance from any training observation to the given Separating plane.

$\boxed{\text{Support vector Machine Fit}}$ — same in neural network also & knn logistic.

```
Index = sample(1:nrow(OJ), 800)
train = OJ[Index, ]
Test = OJ[-Index, ]
```

gradient boost use जानें मत है इते change your variable from factor to numeric.

coefficient of cholostrol = exp(-0.005327)

```
Loocv.err = cv.glm(ad.data, ad.regmod)$delta[1]
```