Machine Learning

100 QUESTIONS

THOMPSON WONG

Introduction

MACHINE LEARNING

THOMPSON WONG

© Thompson Wong

This book is written during the learning of machine learning, it summarized some frequent questions asked by junior or the major points that make people confuse. The original plan is 100 questions, it may be adjusted depends on the practice. Hope it will help you in machine learning field. Enjoy !!!

ABOUT AUTHOR

Thompson Wong
Data Scientists | Machine Learning | Startup
https://kinshing0930.wixsite.com/portfolio

Thompson, experienced with data analysis such as equipment optimization, fault detection and diagnosis, performance evaluation; software development includes analytics framework development, machine learning pipelines and application, RESTful API, optimization algorithm design; research and development for next generation technology; dashboard solution using Grafana.

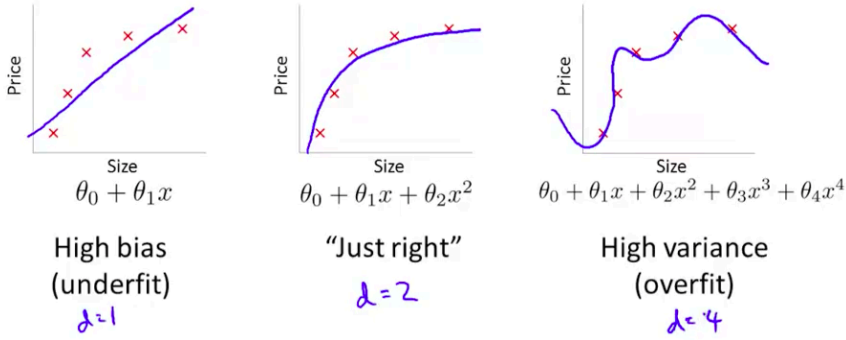
Benefiting from the various working experience in different scale company, a solid experience and good soft skill is developed such as engineering mindset for problem solving, design thinking for the product development, efficiency and effectiveness in software features delivery, leadership to coordinate the works from different stakeholders, communication between client request and team capability.

Communication to digest the needs, design thinking to make a innovation and breakthrough, engineering to solve problem step by step, leadership to coordinate the colleagues and make everyone satisfied in their job assignment.

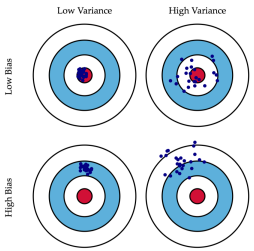
QUESTION 1

BIAS AND VARIANCE

Bias and variance is a basic concept to illustrate the machine learning model performance, like overfitting and under-fitting. High variance also means high uncertainty.



The bias usually represent the underfitting, as insufficient features makes prediction bias in training set. The variance usually represent the overfitting, too much features make machine learning model only perform well in training set, and get fail in newly unseen dataset or test set or validation set. You can comprehend as not generic model.



QUESTION 2

REGULARIZATION

Regularization is an effective things that control the model fitting performance. It is divided by L1 and L2 regularization. The formula is generally written in:

L1 Regularization

$$\text{Cost} = \sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2 + \lambda \sum_{j=0}^{M} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=0}^{N}(y_i - \sum_{j=0}^{M} x_{ij}W_j)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=0}^{M} W_j^2}_{\text{Regularization Term}}$$

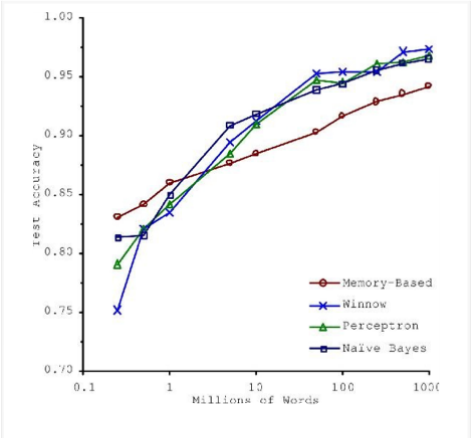
The L1 regularization is usually used in feature selection, L2 regularization is usually used in avoiding overfitting in machine learning. L1 regularization reduce model complexity via feature selection (reducing coefficients parameters/ features amount), L2 regularization reduce complexity by controlling coefficients parameters values.

QUESTION 3

DATA-CENTRIC OR MODEL-CENTRIC?

Data quality is always the major factors affecting the machine learning model performance. Garbage in, garbage out. Spending time for controlling data quality is much better than model algorithm. The existing algorithm is good enough, we all standing on the giant's shoulders.

When you have good enough data quality and quantity, the modelling result will be close to each others in different machine learning algorithms. So don't worry on selecting algorithm or developing new algorithm. Focus on improving data quality, Data-centric AI is greater than model-centric AI. The below figure illustrate the relationship between data quantity, quality and its performance



QUESTION 4

MODEL-BASED VS INSTANCE-BASED LEARNING

Model-based usually knew as supervised learning such as SVM (support vector machine), instance-based learning usually knew as unsupervised learning such as kNN.

In model-based learning, once model is built, the training data can be discarded (e.g. learned weights and bias value in SVM: $y=mx+c$) while instance-based learning use the entire dataset as the model, and looks at the close neighbourhood of the input example in the space of feature vectors and outputs the label that it saw the most often in this close neighbourhood.

Fundamentally, the model-based and instance-based learning has no direct relationship with the terms of data-centric and model-centric. Don't get confused even the terms are similar.

QUESTION 5

DIFFERENCES BETWEEN RF, XGBOOST, LIGHTGBM, GBDT

XGBoost, lightGBM, GBDT, RF are attributed as **ensemble learning***, and boosting as major techniques to improve model performance while random forest adopts bagging method to improve performance.

Generally, the relationship is similar to the iteration of technologies. Therefore, RF -> GBDT -> XGBoost -> LightGBM. I shall create new question to detailedly explain each algorithms.

QUESTION 6

ENSEMBLE LEARNING

The purpose of ensemble learning is that combining multiple base-estimator's prediction result to improve single estimator's robustness and generalization.

The current ensemble learning is divided by two classes. One is strong relationship, serialization is required for result generation. Another one is no strong relationship between estimators, use parallel method to create multiple estimators. The representative of the former is Boosting, and the representative of the latter is Bagging.

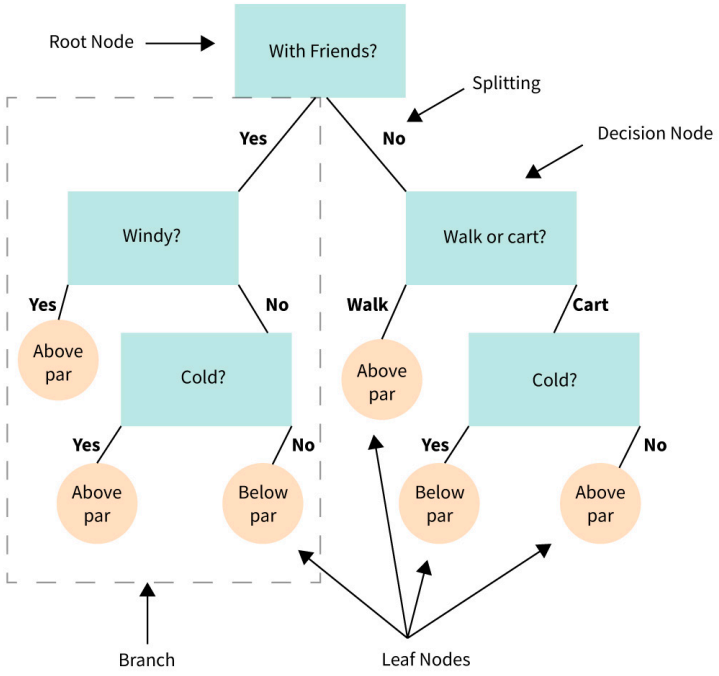
Some terms can also be comprehended as ensemble learning as well such as blending, stacking. People may claim this is different. But no worries, it is the same. Just bear in mind is ok, no need to argue for a long time.

QUESTION 7

DECISION TREE AND RANDOM FOREST

Random forest and decision will be discussed together as they are complementary to each other.

Terms in tree models: root node, leaf node, decision node, leaf cutting, splitting, branch, parent node, child node…



Splitting: process of splitting the root node into multiple node
Leaf node: the node can't be further split
Decision node: a child node will be split for another child node. It is called decision node
Pruning: the process of removing the child node under decision node. It is used to avoid over fitting

Classification and regression tree (CART), the major classes of decision tree model. Classification tree will use "mode" as prediction output. Regression tree will use "average" as prediction output.

The tree model usually use the following method as the evaluation of the splitting performance.
Classification) gini impurity, information gain. The ultimate goal is reducing the uncertainty of the prediction, ensure the probability.
Regression) variance. Lower variance will be selected as the decision node values. It is because lower variance means prediction has lower fluctuant prediction, or relative stable prediction, or not widely distributed results.

The splitting process will take the lowest variance as the decision value in the node.

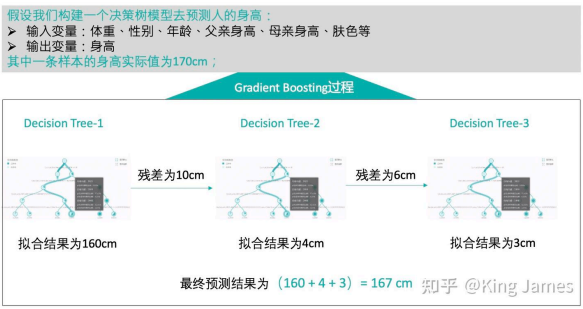
QUESTION 8

GRADIENT BOOSTING DECISION TREE

Gradient boosting decision tree also named as multi additive regression tree. (It is useful, additive telling you it is model use summation.)

The fundamental of gradient boosting decision tree is residuals based learning (fitting the negative gradient). Assuming you have dataset (x_1, y_1), (x_2, y_2), …(x_n, y_n). Then you use a model (linear or whatever like $F(x)$) to fit these data and minimise the loss function. After model fitted, you found its fitting effect is good but potential to advance. (e.g. residuals examples: 0.8 vs 0.9 | 1.4 vs 1.3). The easier way is creating a new model to fit the residuals of $F(x)$, which equals to $y_i - F(x_i)$.

The training dataset will be $(x_1, y_1 - F(x_1))$, $(x_2, y_2 - F(x_2))$, … $(x_n, y_n - F(x_n))$. The target of next estimator is changed to fit the residuals of former estimator. While next estimator can predict the residuals, and you sum it up, the model is accurate. Allow me to reference a picture for easy illustration.



After several iterations, the model will converge finally. Thats why boosting is serialization method, and not support parallel like random forest.

Some may confuse, why GBDT use negative gradient. Loss function shall be minimum when ytrue=ypredicted . If you remember, we have regularization in the cost function. Therefore, when model adopts regularization, it is not able to use residuals for fitting, we shall calculate the gradient of loss function. In most of situation, if just fitting the residuals, the loss go smaller but higher regularization. The cost function may not be reduced. The objective function can be expressed by the gradient, for the purpose of uniform unit scale.

The gradient is equal to the residuals when the loss function is defined as:

$$loss\ function = MSE = (y - F(x))^2$$

And the derivative is $constant * (y - F(x)) = -\frac{\partial J}{\partial F(x)}$, so people always say gradient is residuals (they missed the statement of when loss function is equal to MSE). Regardless of the form of the loss function, each decision tree fits a negative gradient. To be precise, instead of replacing the residual with a negative gradient, when the loss function is a mean square loss, the negative gradient is just the residual, and the residual is only a special case.

GBDT also has high sensitivity to noisy data, as below shown:

y_i	0.5	1.2	2	5*
$F(x_i)$	0.6	1.4	1.5	1.7
$L = (y - F)^2/2$	0.005	0.02	0.125	5.445

GBDT which will have big focus on the fourth data, try to fit the error 5.445 in next estimator.

(More data, will have more precise decision node as node splitting based on the data and its variance after splitting)

QUESTION 9

XGBOOST

XGBoost is advancement of GBDT, it makes several improvements on engineering perspectives. Some common features:

1) XGBoost use 2nd Taylor expansion approximation of loss, and get 1st derivatives and 2nd derivatives
2) XGBoost support multiple base estimators like linear
3) Adopts some method in random forest like columnar sampling
4) Added regularization in cost function to control tree model complexity
5) Shrinkage, equal to learning rate (eta). All leaf node multiply by learning rate to reduce each trees' impact
6) NaN handling, auto calculate the splitting direction
7) Parallel in feature level. Use pre-sorting

I know it is not meaningful to you, so I will skip to the introduction of XGBoost.

XGBoost will iterate all values in all features, find the best splitting in best gain (or differences) of loss changes. The pre-sorting need to keep values of feature and index. Level-wise splitting treat all leaf as the same, XGBoost will still split it even small gain. So the computation cost is high.

The model objective function is expressed as follow:

- Model: assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}$$

- Objective

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Training loss　　Complexity of the Trees

Remember tree model is not suitable to be optimized using SGD (stochastic gradient descent). Unlike linear regression, it has numerical vectors.

- Start from constant prediction, add a new function each time

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\cdots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

New function

Model at training round t　　Keep functions added in previous round

- The prediction at round t is $\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + f_t(x_i)$

This is what we need to decide in round t

$$\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}$$

Goal: find f_t to minimize this

- Consider square loss

$$\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - (\hat{y}_i^{(t-1)} + f_t(x_i))\right)^2 + \Omega(f_t) + const \\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}$$

残差

This is usually called residual from previous round

As the figure shown, the prediction of model in round t is equal to the summation of former (t-1) model's prediction plus f_t .

- Goal $Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant$
  - Seems still complicated except for the case of square loss
- Take Taylor expansion of the objective
  - Recall $f(x + \Delta x) \simeq f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$ 保留二次项
  - Define $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

$$Obj^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$

- *If you are not comfortable with this, think of square loss*

$$g_i = \partial_{\hat{y}^{(t-1)}} (\hat{y}^{(t-1)} - y_i)^2 = 2(\hat{y}^{(t-1)} - y_i) \quad h_i = \partial^2_{\hat{y}^{(t-1)}} (y_i - \hat{y}^{(t-1)})^2 = 2$$

- Compare what we get to previous slide

You will see the specials of XGBoost, objective function keeps Taylor expansion 2nd order.

- Objective, with constants removed

$$\sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

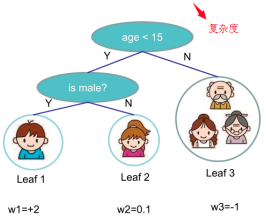
  - where $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$

The complexity of tree is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Number of leaves　　L2 norm of leaf scores

复杂度　　平滑？？？



$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$

QUESTION 10

LIGHTGBM

lightGBM is an improvement based on XGBoost, it features in histogram based decision tree:

1) RAM advantages
2) Computation advantages
3) Leaf-wise splitting
4) Categorial features support
5) Feature parallel
6) Data parallel

As XGBoost, I know it is meaningless to you. I will skip to introduce lightGBM.

Obviously, the memory consumption of the histogram algorithm is (#data* #features * 1Bytes) (because only the value after feature discretization is saved after the feature is bucketed), and the memory consumption of the exact algorithm of xgboost is: (2 * # data * #features* 4Bytes), because xgboost not only saves the original feature value, but also the sequential index of this value. These values need 32-bit floating point numbers to save.

Computational advantages, the pre-sorting algorithm needs to traverse the feature values of all samples when selecting the split features to calculate the split income, the time is (#data), while the histogram algorithm only needs to traverse the bucket, and the time is (#bin)

The histogram of a child node can be obtained by subtracting the histogram of the sibling node from the histogram of the parent node to speed up the calculation.

When splitting discrete features, each value is treated as a bucket, and the gain during splitting is the gain of "whether it belongs to a certain category". Similar to one-hot encoding.

More deeply, like feature parallel and data parallel. Sorry, I am not graduated from CS degree. My computer knowledge is limited. I can’t provide an easy illustration to you guys.

QUESTION 11

COST FUNCTION, LOSS FUNCTION, OBJECTIVE FUNCTION

Loss function is usually a function defined by different metrics. It is about the difference between y_{true}, $y_{predicted}$.

$$residuals = loss = y_{true} - y_{predicted}$$

Cost function is more general. It is the summation of loss function over the training set plus regularization (or penalty)

$$cost\ function = \frac{1}{2m}\sum_{i=1}^{n}(y_{true} - y_{predicted})^2 + \frac{1}{2m}\lambda|x|$$

Objective function is the most general term for any function that optimize during training. It is customized. Like a score of specific training.

Loss function is a part of cost function which is a type of an objective function. So you can know it as any function you want to minimize or maximize called objective function.

Don’t get confused in the terms definition, it is interchanged depends on the context, background… all the purpose is for gradient descent. Right?

QUESTION 12

PREPROCESSING IN MACHINE LEARNING

Preprocessing usually represent the process of sample and split, data preparation (missing values, data types, encoding, imbalance, data augmentation), scale and transform (normalization, transformation, target transformation), feature engineering (extraction, polynomial features, group features, bin features), feature selection (importance, remove multicollinearity, pca analysis, ignore low variance), remove outliers etc…

It this chapter, will more focus on feature scaling as it always make people confused on the topic of differences between standardization and normalization. One thing that is confusing is to refer to confusion. Standardization refers to a clearer term, but normalization sometimes refers to min-max normalization, sometimes refers to Standardization, and sometimes refers to Scaling to unit length.

Some people may be confused in a question, should I use scaling technique before machine learning model training? The answer is yes, but sometime no. it all depends on your algorithm.

For tree based model, you don’t need preprocessing. Tree based model focus on the splitting.

Preprocessing is recommended to be applied before in training data only. The mean and variance are different if you applied preprocessing in training dataset, whole dataset, both dataset. When you apply preprocessing in whole dataset, it will consider all data. To ensure that no “peeking ahead”.Therefore, you are recommended to save the preprocessor of training stage for next prediction purpose.

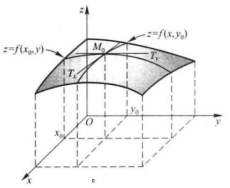
QUESTION 13

CROSS VALIDATION

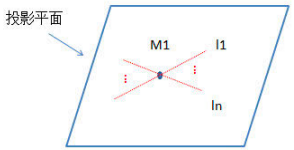
QUESTION 14

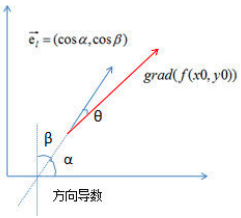
WHY USE GRADIENT IN OPTIMIZATION

The loss only represent the residuals, is scalar type that only includes the magnitude while the gradient represents the vector which includes the magnitude and direction. Therefore, we use gradient to make machine know it is uptrend or downtrend. Combining the learning rate, the next local minimum/maximum can be found. also gradient descent is an general method apply to all optimization question.



To summarize, there are countless directional derivatives of a point in the surface. When the directional derivative is consistent with the gradient direction, the derivative value is the largest, which is equivalent to the point having the fastest change value in the gradient direction. The gradient direction is the direction in which the function value increases the fastest, and the opposite direction of the gradient is the direction in which the function value decreases the fastest.





QUESTION 15

WHAT METRICS SHOULD I USE IN REGRESSION

Common metrics in regression is listed as below:

1) MSE
2) RMSE
3) R2
4) MAE
5) MLE
6) RMLE
7) …

https://mp.weixin.qq.com/s/XlF-6LFXTWIpW-_ke6_DTQ