 ASSIGNMENT - 5

MACHINE LEARNING

**Q1 to Q15 are subjective answer type questions, Answer them briefly.**

1. R-squared or Residual Sum of Squares (RSS) which one of these two is a better measure of goodness of fit model in regression and why?

No, these numbers do not contradict each other. It sounds like you have an understanding of what R2 means: it represents the proportion of the variance in your data which is explained by your model; the closer to one, the better the fit. The residual sum of squares (RSS) is the sum of the squared distances between your actual versus your predicted value.

RSS=∑i=1n(yi−y^i)2

Where yi is a given datapoint and y^i is your fitted value for yi. The actual number you get depends largely on the scale of your response variable. Taken alone, the RSS isn't so informative.

1. What are TSS (Total Sum of Squares), ESS (Explained Sum of Squares) and RSS (Residual Sum of Squares) in regression. Also mention the equation relating these three metrics with each other.

TSS = RSS + ESS

Explained sum of squares (ESS): Also known as the explained variation, the ESS is the portion of total variation that measures how well the regression equation explains the relationship between X and Y.

Residual sum of squares (RSS): This expression is also known as unexplained variation and is the portion of total variation that measures discrepancies (errors) between the actual values of Y and those estimated by the regression equation.

Total sum of squares (TSS): It is a quantity that appears as part of a standard way of presenting results of such analyses. For a set of observations {\displaystyle y\_{i},i\leq n}{\displaystyle y\_{i},i\leq n}, it is defined as the sum over all squared differences between the observations and their overall mean {\displaystyle {\bar {y}}}{\bar {y}}.

1. What is the need of regularization in machine learning?

In general, regularization means to make things regular or acceptable. This is exactly why we use it for applied machine learning. In the context of machine learning, regularization is the process which regularizes or shrinks the coefficients towards zero. In simple words, regularization discourages learning a more complex or flexible model, to prevent overfitting.

1. What is Gini–impurity index?

Gini index or Gini impurity measures the degree or probability of a particular variable being wrongly classified when it is randomly chosen. But what is actually meant by ‘impurity’? If all the elements belong to a single class, then it can be called pure. The degree of Gini index varies between 0 and 1, where 0 denotes that all elements belong to a certain class or if there exists only one class, and 1 denotes that the elements are randomly distributed across various classes. A Gini Index of 0.5 denotes equally distributed elements into some classes.

1. Are unregularized decision-trees prone to overfitting? If yes, why?

Yes, unregularized decision-trees prone to overfitting. Overfitting happens when any learning processing overly optimizes training set error at the cost test error. While it’s possible for training and testing to perform equality well in cross validation, it could be as the result of the data being very close in characteristics, which may not be a huge problem. In the case of decision tree’s they can learn a training set to a point of high granularity that makes them easily overfit. Allowing a decision tree to split to a granular degree, is the behavior of this model that makes it prone to learning every point extremely well — to the point of perfect classification — ie: overfitting.

1. What is an ensemble technique in machine learning?

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. The goal of any machine learning problem is to find a single model that will best predict our wanted outcome. Rather than making one model and hoping this model is the best/most accurate predictor we can make, ensemble methods take a myriad of models into account, and average those models to produce one final model.

1. What is the difference between Bagging and Boosting techniques?

Bagging is a method of merging the same type of predictions. Boosting is a method of merging different types of predictions.

Bagging decreases variance, not bias, and solves over-fitting issues in a model. Boosting decreases bias, not variance.

In Bagging, each model receives an equal weight. In Boosting, models are weighed based on their performance.

Models are built independently in Bagging. New models are affected by a previously built model’s performance in Boosting.

1. What is out-of-bag error in random forests?

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows: Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree. Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.

1. What is K-fold cross-validation?

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model. It is a popular method because it is simple to understand and because it generally results in a less biased or less optimistic estimate of the model skill than other methods, such as a simple train/test split.

1. What is hyper parameter tuning in machine learning and why it is done?

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data.The objective function takes a tuple of hyperparameters and returns the associated loss.Cross-validation is often used to estimate this generalization performance.

1. What issues can occur if we have a large learning rate in Gradient Descent?

The learning rate hyperparameter controls the rate or speed at which the model learns. Specifically, it controls the amount of apportioned error that the weights of the model are updated with each time they are updated, such as at the end of each batch of training examples. Given a perfectly configured learning rate, the model will learn to best approximate the function given available resources (the number of layers and the number of nodes per layer) in a given number of training epochs (passes through the training data). Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train. At extremes, a learning rate that is too large will result in weight updates that will be too large and the performance of the model (such as its loss on the training dataset) will oscillate over training epochs. Oscillating performance is said to be caused by weights that diverge (are divergent). A learning rate that is too small may never converge or may get stuck on a suboptimal solution.

1. Can we use Logistic Regression for classification of Non-Linear Data? If not, why?

No, we can’t use logistic regression for classification of Non-Linear data. Logistic regression is known and used as a linear classifier. It is used to come up with a hyperplanein feature space to separate observations that belong to a class from all the other observations that do notbelong to that class. The decision boundary is thus linear. Robust and efficient implementations are readily available (e.g. scikit-learn) to use logistic regression as a linear classifier.

1. Differentiate between Adaboost and Gradient Boosting.

|  |  |  |
| --- | --- | --- |
| 1. S.No | Adaboost | Gradient Boost |
| 1 | An additive model where shortcomings of previous models are identified by high-weight data points. | An additive model where shortcomings of previous models are identified by the gradient. |
| 2 | The trees are usually grown as decision stumps. | The trees are grown to a greater depth usually ranging from 8 to 32 terminal nodes. |
| 3 | Each classifier has different weights assigned to the final prediction based on its performance. | All classifiers are weighed equally and their predictive capacity is restricted with learning rate to increase accuracy. |
| 4 | It gives weights to both classifiers and observations thus capturing maximum variance within data. | It builds trees on previous classifier’s residuals thus capturing variance in data. |

1. What is bias-variance trade off in machine learning?

The bias–variance tradeoff is the property of a model that the variance of the parameter estimates across samples can be reduced by increasing the bias in the estimated parameters. The bias-variance tradeoff is a central problem in supervised learning. Ideally, one wants to choose a model that both accurately captures the regularities in its training data, but also generalizes well to unseen data. Unfortunately, it is typically impossible to do both simultaneously. High-variance learning methods may be able to represent their training set well but are at risk of overfitting to noisy or unrepresentative training data. In contrast, algorithms with high bias typically produce simpler models that may fail to capture important regularities (i.e. underfit) in the data.

1. Give short description each of Linear, RBF, Polynomial kernels used in SVM.

Linear= Linear Kernel is used when the data is Linearly separable, that is, it can be separated using a single Line. It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set. One of the examples where there are a lot of features, is Text Classification, as each alphabet is a new feature. So we mostly use Linear Kernel in Text Classification.

RBF = Gaussian RBF(Radial Basis Function) is another popular Kernel method used in SVM models for more. RBF kernel is a function whose value depends on the distance from the origin or from some point. RBF kernels are the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian distribution. The RBF kernel function for two points X₁ and X₂ computes the similarity or how close they are to each other.

Polynomial = Polynomial kernel is a kernel function commonly used with support vector machines (SVMs) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables, allowing learning of non-linear models.Intuitively, the polynomial kernel looks not only at the given features of input samples to determine their similarity, but also combinations of these. In the context of regression analysis, such combinations are known as interaction features. The (implicit) feature space of a polynomial kernel is equivalent to that of polynomial regression, but without the combinatorial blowup in the number of parameters to be learned. When the input features are binary-valued (booleans), then the features correspond to logical conjunctions of input features.