MACHINE LEARNING – WORKSHEET 3

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

   The Support Vector Machine is a supervised learning algorithm mostly used for classification but it can be used also for regression. The main idea is that based on the labelled data (training data) the algorithm tries to find the optimal hyperplane which can be used to classify new data points. In two dimensions the hyperplane is a simple line.

   4. The Gini impurity measure is one of the methods used in decision tree algorithms to decide the optimal split from a root node, and subsequent splits.

Let Ginx represent the gini index.

$$Ginx = \bar{p_1} \cdot (1 - p_1) + p_2 \cdot (1 - p_2)$$
$$equivalently,$$
$$Ginx = 2 \cdot p_1 p_2$$

Where p1, p2 are class 1 , 2 probabilities, respectively.

Note: p1 + p2 =1

This is not complete yet. The equation above will give us the gini impurity measure for a sub split, but we would like to know the gini impurity measure for the entire split (because the data will be split to the left and right). Therefore, we will need to weigh them accordingly.

$$WeightedGinx = P_L \cdot (2 \cdot p_{L1} p_{L2}) + P_R \cdot (2 \cdot p_{R1} p_{R2})$$

Where P_L is the proportion of the split that goes to the left side and p_L1 is analogous to p1 for the left split(same for P_R).

6. **Ensemble methods** is a machine learning technique that combines several base models in order to produce one optimal predictive model. To better understand this definition lets take a step back into ultimate goal of machine learning and model building. This is going to make more sense as I dive into specific examples and why Ensemble methods are used.

*BAGG*ing, or *B*ootstrap *AGG*regating. **BAGG**ing gets its name because it combines *B*ootstrapping and *Agg*regation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor.

**Random Forest** Models. Random Forest Models can be thought of as **BAGG**ing, with a slight tweak. When deciding where to split and how to make decisions, BAGGed Decision Trees have the full disposal of features to choose from. Therefore, although the bootstrapped samples may be slightly different, the data is largely going to break off at the same features throughout each model. In contrary, Random Forest models decide where to split based on a random selection of features. Rather than splitting at similar features at each node throughout, Random Forest models implement a level of differentiation because each tree will split based on different features. This level of differentiation provides a greater ensemble to aggregate over, ergo producing a more accurate predictor.

9. 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.

The general procedure is as follows:

1. Shuffle the dataset randomly.
2. Split the dataset into k groups
3. For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
4. Summarize the skill of the model using the sample of model evaluation scores

10. Hyperparameter tuning works by running multiple *trials* in a single training job. Each trial is a complete execution of your training application with values for your chosen hyperparameters, set within limits you specify. The AI Platform Training training service keeps track of the results of each trial and makes adjustments for subsequent trials. When the job is finished, you can get a summary of all the trials along with the most effective configuration of values according to the criteria you specify.

Hyperparameter tuning requires explicit communication between the AI Platform Training training service and your training application. Your training application defines all the information that your model needs. You must define the hyperparameters (variables) that you want to adjust, and a target value for each hyperparameter.

To learn how AI Platform Training uses Bayesian optimization for hyperparameter tuning, read the blog post named Hyperparameter Tuning in Cloud Machine Learning Engine using Bayesian Optimization.

In addition to Bayesian optimization, AI Platform Training optimizes across hyperparameter tuning jobs. If you are doing hyperparameter tuning against similar models, changing only the

objective function or adding a new input column, AI Platform Training is able to improve over time and make the hyperparameter tuning more efficient.

Hyperparameter tuning optimizes a single target variable, also called the hyperparameter metric, that you specify. The accuracy of the model, as calculated from an evaluation pass, is a common metric. The metric must be a numeric value, and you can specify whether you want to tune your model to maximize or minimize your metric.

When you start a job with hyperparameter tuning, you establish the name of your hyperparameter metric. This is the name you assign to the scalar summary that you add to your training application.

The default name of the metric is `training/hptuning/metric`. We recommend that you assign a custom name. The only functional difference is that if you use a custom name, you must set the `hyperparameterMetricTag` value in the HyperparameterSpec object in your job request to match your chosen name.