School of Computing and Information Systems
The University of Melbourne
COMP30027 Machine Learning (Semester 1, 2021)

Sample Solutions: Week 9

1. What is the difference between “model bias” and “model variance”?

   Model Bias:

   – Model bias is the propensity of a classifier to systematically produce the same errors; if it doesn’t produce errors, it is unbiased; if it produces different kinds of errors on different instances, it is also unbiased. (An example of the latter: the instance is truly of class A, but sometimes the system calls it B and sometimes the system calls it C.)

   – Note that this makes more sense in a regression context, where we can sensibly measure the difference between the prediction and the true value. In a classification context, these can only be “same” or “different”.

   – Consequently, a typical interpretation of bias in a classification context is whether the classifier labels the test data in such a way that the distribution of predicted classes systematically doesn’t match the distribution of actual classes. For example, "bias towards the majority class", when the model predicts too many instances as the majority class.

   Model variance is the propensity of a classifier to produce different classifications using different training sets (randomly sampled from the same population). It is a measure of the inconsistency of the classifier, from training set to training set.

   (i). Why is a high bias, low variance classifier undesirable?

   In short, because it’s consistently wrong. Using the other interpretation: the distribution of labels predicted by the classifier is consistently different to the distribution of the true labels; this means that it must be making mistakes.

   (ii). Why is a low bias, high variance classifier (usually) undesirable?

   This is less obvious – it’s low bias, so that it must be making a bunch of correct decisions. The fact that it’s high variance means that not all of the predictions can possibly be correct (or it would be low-variance!) — and the correct predictions will change, perhaps drastically, as we change the training data.

   One obvious problem here is that it’s difficult to be certain about the performance of the classifier at all: we might estimate its error rate to be low on one set of data, and high on another set of data.

   The real issue becomes more obvious when we consider the alternative formulation: the low bias means that the distribution of predictions matches the distribution of true labels; however, the high variance means that which instances are getting assigned to which label must be changing every time.

   This suggests the real problem — namely, that what we have is the second kind of unbiased classifier: one that makes different kinds of errors on different training sets, but always errors; and not the first kind: one that is usually correct.

2. Describe how validation set, and cross-validation can help reduce overfitting?

Machine learning models usually have one or more (hyper)parameters that control for model complexity, the ability of model to fit noise in the training set. In a practical application, we need to determine the values of such parameters, and the principal objective in doing so is usually to achieve the best predictive performance on new data. Furthermore, as well as finding the appropriate values for complexity parameters within a given model, we may wish to consider a range of different types of model in order to find the best one for our particular application.

We know that the performance on *training data* is not a good indicator of predictive performance on unseen data because of overfitting. If data is plentiful, then one approach is simply to use some of the available data to train a range of models, or a given model with a range of values for its complexity parameters, and then to compare them on independent data, sometimes called a *validation set*, and select the one having the best predictive performance. If the model design is iterated many times using a limited size data set, then some overfitting to the validation data can occur and so it may be necessary to keep aside a *third test set* on which the performance of the selected model is finally evaluated.

In many applications, however, the supply of data for training and testing will be limited, and in order to build good models, we wish to use as much of the available data as possible for training. However, if the validation set is small, it will give a relatively noisy estimate of predictive performance. One solution to this dilemma is to use *cross-validation*.

3. Why *ensembling* reduces model variance?

We know from statistics that averaging reduces variance. If $Z_1, \ldots, Z_n$ are i.i.d random variables:

$$Var(\frac{1}{N}\sum_i Z_i) = \frac{1}{N} Var(Z_i)$$

So, the idea is that if several models are averaged, the model variance decreases without having an effect on bias. The problem is that there is only one training set, so how do we get multiple models? The answer to this problem is ensembling. Ensembling creates multiple models by creating multiple training sets from one training set by using bootstrap (e.g., bagging, random forests), or by training multiple learning algorithms (e.g., stacking). The predictions of the individual are then combined (averaged) to reduce final model variance.

4. Imagine you are given a dataset from the university's library, and your job is to build a classification model that classify students based on the list of books that they borrowed.

The dataset includes the list of books available in the library (columns) and the students who borrowed them (rows), and the ranking for each item (ranking value is between 0–5, 0 if the book was not borrowed and 1–5 indicates the student's interest). The metadata for the books (e.g. titles) are not readily available to us, we just have the book IDs (e.g. Book #i). The dataset also includes the students' field of study (in total there are 10 fields), which can be used for the classification task. Answer the following questions, considering that there are 500,000 students and 100,000 books in this dataset.

Student ID	Book #1	⋯	Book #100,000	Label (Field of Study)
Student # 1	3	⋯	2	Computer science
Student # 2	5	⋯	0	Biology
⋮	⋮	⋮	⋮	⋮
Student # 500,000	1	⋯	4	Mathematic

(i). Consider the following supervised machine learning methods, and for each one, explain why it would be appropriate or inappropriate to use for this problem:

i. Naïve Bayes

Too many attributes, most probabilities based on numeric values will be essentially random. Naïve Bayes is oversensitive to redundant and/or irrelevant attributes.

ii. k-NN

Too many dimensions, similarities are mostly meaningless. Also due to high dimension of the instances (100,000), calculating the similarity would be computationally heavy.

iii. Decision Tree

Too many attributes. With too many nodes Decision Tree is in danger of overfitting.

(ii). Would "feature selection" be useful here? Explain why, by referring to a single machine learning method.

Feature selection is expected to improve any of the above approaches.

The distances used in K-NN become meaningless in high-dimensional space (intuitively, everything is far away from everything). So, feature selection is very important.

Naive Bayes is slightly more robust to many / irrelevant features, but since it does not have an embedded feature weighting mechanism (unlike e.g., logistic regression) it is expected to improve.

Decision trees are prone to overfitting, and we know that feature selection can help in this case. Decision trees are, in fact, have an embedded method for feature selection -- they perform selection as part of the model. For example, you can prune the decision tree ('chop off' branches with low IG) to rid yourself of unpredictive features and get a more robust model.

(iii). Explain how you would evaluate the effectiveness of your system: you should briefly describe an evaluation strategy and an evaluation metric that are suitable for this data. What might be an example of a baseline?

- Evaluation strategy: Cross-validation - partition the data into M (approximately) equal size partitions, train the system M times and the average performance is computed across the M runs.
- Evaluation metric: F1-score or accuracy
- Baseline: zero-R - classify all instances according to the most common class in the training data. OR Random Baseline - randomly assign a class to each test instance