**How to Create a Bagging Ensemble in Neural Nets**

Ensemble learning are methods that combine the predictions from multiple models.

It is important in ensemble methods that the models that comprise the ensemble are reasonably good, making different prediction errors. Using an ensemble of predictions that are reasonable can be combined to result in a prediction that is at once more stable and better than a prediction from any individual member model.

One way to achieve differences between models is to train each model on a different subset of the available training data. Models are trained on different subsets of the training data naturally through the use of resampling methods such as cross-validation and the bootstrap, designed to estimate the average performance of the model generally on unseen data. The models used in this estimation process can be combined in what is referred to as a resampling-based ensemble, such as a cross-validation ensemble or a bootstrap aggregation (or bagging) ensemble.

In this case we develop a collection of different resampling-based ensembles for deep learning neural network models.

The goal is to learn:

* How to estimate model performance using random-splits and develop an ensemble from the models.
* How to estimate performance using 10-fold cross-validation and develop a cross-validation ensemble.
* How to estimate performance using the bootstrap and combine models using a bagging ensemble.

.

**Overview**

Bagging estimate prediction is divided into six parts; they are:

1. Data Resampling Ensembles
2. Multi-Class Classification Problem
3. Single Multilayer Perceptron Model
4. Random Splits Ensemble
5. Cross-Validation Ensemble
6. Bagging Ensemble

**Data Resampling Ensembles**

Combining the predictions from multiple models can result in more stable predictions, and in some cases, predictions that are better than any of the contributing models.

Effective ensembles require members that disagree. Each member must have “skill” (*i.e*. perform better than random chance), but ideally the members of the collection of models all perform well in different ways. Put another way, we prefer ensemble members to have low correlation in their predictions, or prediction errors.

One approach to encourage differences among ensembles is to use the same learning algorithm on different training datasets. This can be achieved by repeatedly resampling a training dataset and then use this resampled set to train a new model. Multiple models are fit using slightly different perspectives on the training data and, in turn, make different errors. In this manner more stable and better predictions are typically produced when the ensemble of model results is combined.

The approach is basically to generate a set of models by resampling the training set and generate a prediction from the ensemble of predictions from these models.

A benefit of this approach is that resampling the training set makes use of subsets that may have slightly different model characteristics. Then data that are not used to fit the model can be used as a test dataset to estimate the generalization error of the chosen model configuration.

There are three popular resampling methods that we could use to create a resampling ensemble; they are:

* **Random Splits**. The dataset is repeatedly sampled with a random split of the data into train and test sets.
* **k-fold Cross-Validation**. The dataset is split into k equally sized folds, k models are trained and each fold is given an opportunity to be used as the holdout set where the model is trained on all remaining folds.
* **Bootstrap Aggregation**. Random samples are collected with replacement and examples not included in a given sample are used as the test set.

The most widely used resampling ensemble method currently is the bootstrap aggregation-- bagging. Resampling with replacement allows more difference in the training dataset, biasing the model and, in turn, resulting in differences between the predictions of the resulting models.

Resampling ensemble models are not with cost/assumptions: *viz.,*

* That a robust estimate of model performance on unseen data is required (make a prediction on ‘as yet unseen’ data); if not, then a single train/test split can be used.
* That there is a potential for a lift in performance using an ensemble of models; if not, then a single model fit on all available data can be used.
* That the computational cost of fitting more than one neural network model on a sample of the training dataset is not prohibitive; if not, all resources should be put into fitting a single model.

Neural network models are remarkably flexible; typically the lift in performance provided by a resampling ensemble may not always provide improved predictions. It is possible that an individual model trained on available data can perform well. An example of this is predicting species in the Iris data set.

The best application of a resampling ensemble occurs when there is a requirement for a robust estimate or prediction, and multiple models yield different results (as in the Boston ‘median value of a home’ estimate. As long as models created using the neural net yield different final predictions, bagging is appropriate. Put another way, a single final model fit on all available training data doesn’t do that well.

Next we’ll consider an example.

**Multi-Class Classification Problem**

We will use a small multi-class classification problem as the basis to demonstrate a model resampling ensembles. This is motivated by the class project for deterring an approach to ‘how to advertise’ for selling used cars.

The advertising problem is a good candidate in that it is non-trivial and allows a neural network model to find a variety of different “*good enough*” candidate solutions.

**Single Multilayer Perceptron Model**

We will define a Multilayer Perceptron neural network, or MLP, that learns the problem reasonably well.

The problem is a simple classification problem: predict ‘sale / no sale’ based on characteristics of buyers of used. We split the dataset into training and test sets, then use the test set both to evaluate the performance of the model and to plot a histogram of its performance. We will use 90% of the data for training and 10% for the test set. We choose a large split because the data are fairly noisy, and a well-performing model requires as much data as possible to learn the complex classification function.

The sales NN model expects samples with two between two and six input variables. The model then has a single hidden layer with xx nodes, and then an output layer with 2 nodes to predict that a potential customer with the given set of input parameters would/would not buy a used car at or above the selected sales $$.

Because the problem is a class neural net assignment, we’ll limit the number separate models to no more than 10. For each of the re-sampled models, we evaluate the model on the test set. The original data are broken into a 90/10 train and test set. This train set is resampled with replacement, and a neural net model produced. Each of the re-sampled models is evaluated using the re-sampled training set and the fixed test set. We summarize the prediction results of the original model, and the prediction results of the bagged predictions. From these recommendations for advertising are made! BTW, your specific results will vary from other’s results by design!. The purpose of the project is to get experience with neural nets and bagging.

The chosen split of the dataset into train and test sets means that the test set is small and not representative of the broader problem. In turn, performance on the test set is not representative of the model; in this case, it is optimistically biased.

**Random Splits Ensemble**

The instability of the model and the small test dataset mean that we don’t really know how well this model will perform on new data in general. That’s OK in this case.

One could use a simple resampling method of repeatedly generating new random splits of the dataset into train and test sets and fit the new models. Calculating the average of the performance of the model across each split would give a better estimate of the model’s generalization error.

Bagging basically combines multiple models trained on the random splits with the expectation that performance of the ensemble is likely to be more stable (and better) than the average single model.

In this example, we will limit the number of splits, and in turn, the number of fit models to 10. After fitting and evaluating the models, we can estimate the expected performance of a given model with the chosen configuration for the domain (input variables). Limiting our collection of re-sampled models to 10 is arbitrary; how many models would product the best result is unknown. It is likely that there is a point of diminishing returns, after which the addition of further models derived by re-sampling the data set no longer changes the performance of the ensemble.

A rough idea of ‘performance improvement’ can be estimated by plotting the probability of correct classification on the test set using ensemble sizes from 1 to 10. More models in the ensemble should result in better predictions.

Further, we save the probability of correct classification for each model (on the holdout dataset) and calculate the average of these scores to get an approximate true performance of the chosen model on the prediction problem.

**Cross-Validation Ensemble**

A problem with repeated random splits as a resampling method for estimating the average performance of model is that it tends to be overly optimistic.

A model validation approach designed to be less optimistic and is widely used as a result is the k-fold-cross-validation method.

K-fold-cross-validation tends to be less biased because each example in the dataset is only used one time in the test dataset to estimate model performance, unlike random train-test splits where a given example may be used to evaluate a model many times.

The procedure has a single parameter called k that refers to the number of groups into which a given data sample is split. (Ending a sentence with a preposition is something up with which I will not put.) The average of the scores of each model provides a less biased estimate of model performance. A typical value for k is 10.

Because neural network models are computationally expensive to train, it is common to use the best performing model during cross-validation as the final model.

Alternately, the resulting models from the cross-validation process could be combined to provide a cross-validation ensemble that is likely to have better performance on average than a given single model. We will not do this.

K-fold-cross validation is based on taking ‘k’ splits of the data set. The train set is broken into k (disjoint) sets. A neural net model is produced for each of the k sets and the probability of correct classification for each of the models is found, using the same test set. Based on these 10 model results, (one for each fold), the average of the scores can be used to estimate the expected performance of the neural net. One could plot the ensemble error using from 1 to 10 of these models, as indicated above. Plot the number of models in the ensemble on the x-axis and the probability of correct classification on the y-axis. When one model is used plot the probability of correct classification across the test set; when two models are combined, plot the probability of correct classification across the test set, and so on.

**Summary: Bagging Ensemble**

A limitation of random splits and k-fold cross-validation from the perspective of ensemble learning is that the models are very similar.

Bootstrap is a statistical technique for estimating quantities about a population by averaging estimates from multiple resampling sets from the collected data sample.

Bootstrap samples in neural nets are constructed by drawing observations from a large data set, one at a time and returning the sample to the data sample so that it may be chosen again. This allows a given observation to be included in a given small sample more than once. This approach to sampling is called sampling with replacement.

The bootstrap is a robust method for estimating model performance. It does suffer a little from an optimistic bias, but is often almost as accurate as k-fold cross-validation in practice.