Overview

The purpose of this assignment is to obtain an understanding of experimental settings used to organize data modeling problems (i.e., regression or classification). We will first dive deep into methods to split data into sets. This will provide us with the knowledge needed to design experiments properly.

Data Splits: Train, Validation, and Test Sets

A validation set is a sample of data held back during training to find the "best" hyper-parameters for the model via tuning values and evaluating this held-out set. Validation sets, like test sets, are held back during the training phase (i.e., training the model). The two differ in that the test is used to give an unbiased estimate of the performance of the final model to compare against other models: the validation is used to compare the same model across different hyper-parameters, whereas the test is used to compare different models that are assumed to have optimal hyper-parameters set). Thus, validation and test sets are used to compare and select at the hyperparameter level and model-level, respectfully.

There is much confusion in applied machine learning about what a validation dataset is and how it differs from a test dataset.

After this assignment, you should be able to define train, test, and validation sets clearly. Additionally, you should know how the data of your term project should be split.

In summary, we will cover:

- The definitions of train, test, and validation sets as seen by experts in fields based on data analysis.
- The difference between validation and test sets in practice.
- Processes that will lead to the best use of validation and test sets during model evaluation.

Data splits: As seen by the experts

To develop a deeper understanding, let's first understand how the experts view the concept of validation. For this, we will review snippets from several of the top machine learning and statistical modeling texts and references. During which, keep an open mind, as subtle differences may exist.

Generally speaking: "Validation set" is used interchangeably with "Test set" and is used to reference a sample of data held back during training. Why? Well evaluating a model on samples used during training would yield a biased score, whereas an evaluation on a set the was held out during training produces an unbiased estimate of performance. Typically, we call this the train-test split approach to algorithm (or model) evaluation.

"*Suppose we would like to estimate the test error associated with fitting a particular statistical learning method on a set of observations. The validation set approach is a very simple strategy for this task. It randomly divides the available set of observations into two parts, a training set and a validation set or hold-out set. The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set. The resulting validation set error rate — typically assessed using MSE in the case of a quantitative response—provides an estimate of the test error rate.*"

– Gareth James, et al., Page 176, An Introduction to Statistical Learning: with Applications in R, 2013.

Next, examine the concept shared in Kuhn and Johnson's excellent text "Applied Predictive Modeling". The authors point out that the final model evaluation must be performed on a held-out set that has yet to be used, whether during training or tuning the model's parameters.

"*Ideally, the model should be evaluated on samples that were not used to build or fine-tune the model, so that they provide an unbiased sense of model effectiveness. When a large amount of data is at hand, a set of samples can be set aside to evaluate the final model. The "training" data set is the general term for the samples used to create the model, while the "test" or "validation" data set is used to qualify performance.*"

– Max Kuhn and Kjell Johnson, Page 67, Applied Predictive Modeling 2013.

Traditionally, the data split used for evaluating model performance was called the test set. Russell and Norvig reinforce the importance of keeping the test set completely separate in their seminal work on AI. The authors refer to the user information of from the test set in any manner as "peeking". The authors go as far as to suggest that the test set be locked away until all model tuning is complete, i.e., "blind testing".

"*Peeking is a consequence of using test-set performance to choose a hypothesis and evaluate it. The way to avoid this is to hold the test set out—lock it away until you are completely done with learning and wish to obtain an independent evaluation of the final hypothesis. (And then, if you don't like the results . . . you have to obtain, and lock away, a completely new test set if you want to go back and find a better hypothesis.)*"

– Stuart Russell and Peter Norvig, page 709, Artificial Intelligence: A Modern Approach, 2009 (3rd edition).

The take-home here is that Russell and Norvig take it a step further by proposing that a training set used to fit the model can be further split into a training set and a validation set. This subset of the training set is called the validation set, which can then be used to get an early estimate of model performance.

"*If the test set is locked away, but you still want to measure performance on unseen data to select a good hypothesis, then divide the available data (without the test set) into a training set and a validation set.*"

– Stuart Russell and Peter Norvig, page 709, Artificial Intelligence: A Modern Approach, 2009 (3rd edition).

This definition of a validation set can be seen across many other seminal texts in the field. A good (and older) example is the glossary of terms in Ripley's book "Pattern Recognition and Neural Networks." Specifically, training, validation, and test sets are defined as follows:

"

1. *Training set: A set of examples used for learning, that is, to fit the classifier's parameters.*
2. *Validation set: A set of examples is used to tune a classifier's parameters, for example, to choose the number of hidden units in a neural network.*
3. *Test set: A set of examples is used only to assess a fully-specified classifier's performance.*

"

– Brian Ripley, page 354, Pattern Recognition and Neural Networks , 1996.

The above definitions are the recommended usages.

A good example that these definitions are canonical is their reiteration in the famous Neural Network FAQ. In addition to reiterating Ripley's glossary definitions, it goes on to discuss the common misuse of the terms "test set" and "validation set" in applied machine learning.

"The literature on machine learning often reverses the meaning of "validation" and "test" sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research. The crucial point is that a test set, by the standard definition in the NN [neural net] literature, is never used to choose among two or more networks, so that the error on the test set provides an unbiased estimate of the generalization error (assuming that the test set is representative of the population, etc.)."

– Subject: What are the population, sample, training set, design set, validation set, and test set?

Definitions of Train, Validation, and Test Datasets

Let's now define these three terms without ambiguity.

Training Set: The sample of data used to fit the model.

Validation Set: The sample of data used to provide an unbiased evaluation of a model fit on the training set while tuning model hyper-parameters. The evaluation becomes more biased as performance on the validation set is incorporated into the model configuration.

Test Set: Data samples used to provide an unbiased evaluation of a final model fit on the training dataset. We can make this concrete with a pseudocode sketch:

```
# split data
data = ...
train, validation, test = split(data)

# tune model hyper-parameters
parameters = ...
for params in parameters:
        model = fit(train, params)
        performance = evaluate(model, validation)

# evaluate the final model for comparison with other models
model = fit(train)
skill = evaluate(model, test)
```

Thus, the "validation set" is a subset of the "training set"; the "test set" is a separate partition of the data and should be considered a different dataset entirely (see Figure 1).
Some additional clarifying notes:

1. The validation set may also play a role in other forms of model preparation, such as feature selection.
2. The final model could be fit on the aggregate of the training and validation sets and then "blindly" evaluated on the test set for the final score to report.

Before splitting, consider the following questions:

1. What if one class dominates the first 50% of the data? $\longrightarrow$ shuffle your data. Be careful that it is done fairly. Furthermore, if benchmarks are being compared, mimic what was done to generate the original results.
2. What if the class are not evenly distributed $\longrightarrow$ stratified k-fold, under-sampling, or oversampling.

Make sure that all experiments are reproducible. Also, all steps are documented, with random seed values set and recorded.
Next, let's discuss split ratios.

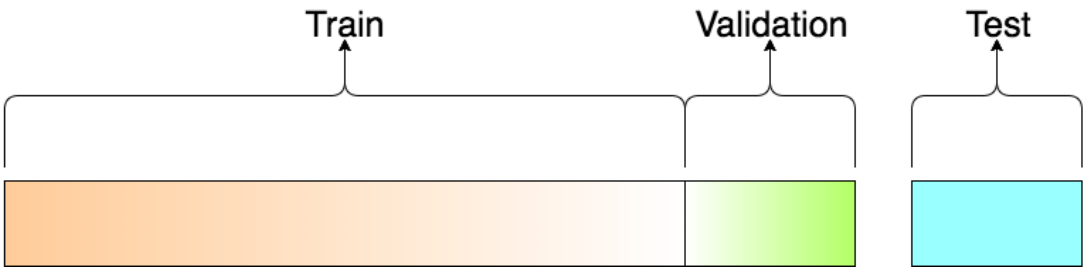

Figure 1: Depiction of data splits. First, the *test* set is split off and set aside, which is not touched until after training. The remaining training data is then split into the *train* set and the *val* set, which are used to find optimal hyper-parameters and to test during training. Often, there is no access to the test set (e.g., a competition with automatic scoring will often only release the *test* set only for a small time block at the end, and the labels might never be released so that the competition can repeat in the future).

Split ratio

A perfectly fair question could pertain to the specific ratios for train, validation, and test sets.
Well, the answer to this question mainly depends on 2 things: (1) the total number of samples in your data and (2) the type of model being trained.

Some models need big data to train. In this case, you would optimize with larger training sets. Models with few hyper-parameters typically are easier to validate and tune. Thus, the size of the validation set can be reduced.

Contrary to this, models made-up of many hyper-parameters demand a large validation set– although cross-validation should be considered. A validation set is unnecessary for models with no hyper-parameters or ones that cannot be easily tuned!

Like most things in machine learning, the train-test-validation split ratio varies on a case-by-case basis. One's judgment improves with more experience (i.e., building more models). A typical split is 70% train and 30% validation, but it varies for different cases.

Validation Set Is Not Enough: k-Fold Cross Validation

There are other ways of assessing models unbiasedly and on unseen data.
One popular technique for tuning a model's hyper-parameter is to use k-fold cross-validation instead of a separate validation set as considered up to this point (See Figure 2).
Kuhn and Johnson have a section called "Data Splitting Recommendations" in which the authors lay out the limitations of using a sole "test set" (or "validation set"):

"*As previously discussed, there is a strong technical case to be made against a single, independent test set:*

- *A test set is a single evaluation of the model and cannot characterize the uncertainty in the results.*
- *Proportionally large test sets divide the data, increasing bias in the performance estimates.*
- *With small sample sizes:*
- *The model may need every possible data point to determine model values adequately.*

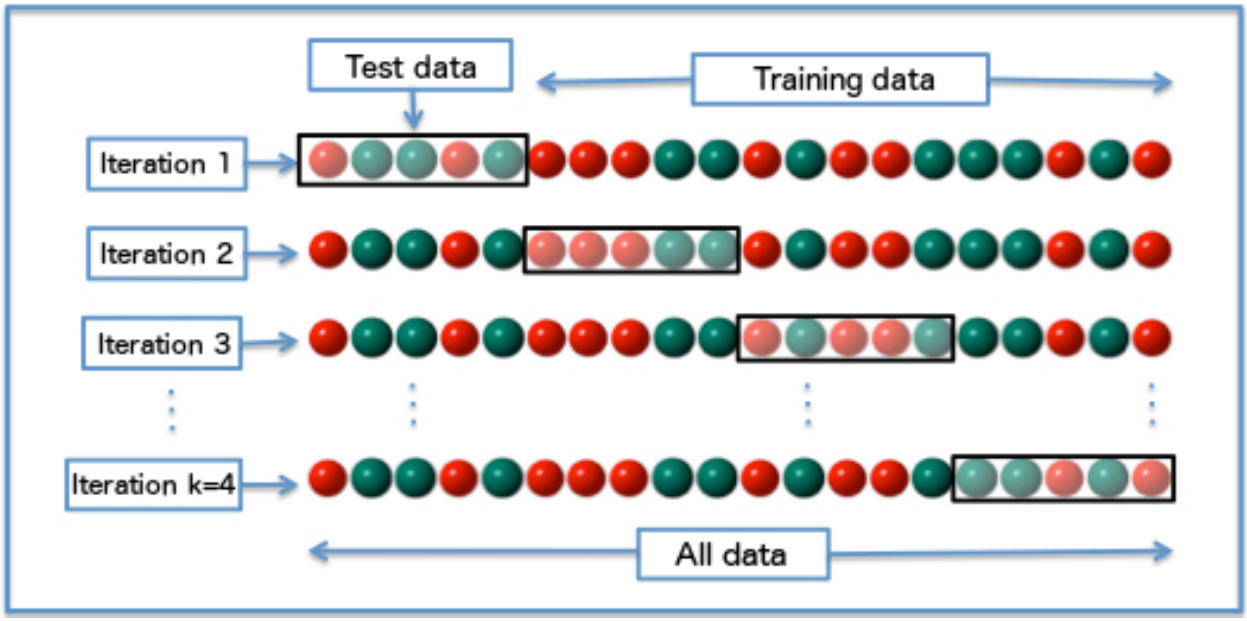

Figure 2: Visualization of k-Fold cross-validation (i.e., k=4). Note that *All data* in the graphic represents all training data, as the *test* set would be set aside from this.

- *The uncertainty of the test set can be considerably large to the point where different test sets may produce very different results.*
- *Resampling methods can reasonably predict how well the model will perform on future samples.*
- Max Kuhn and Kjell Johnson, Page 78, Applied Predictive Modeling 2013.

They recommend using 10-fold cross-validation for smaller datasets to obtain the performance estimate's desirable low bias and variance properties. The authors recommend the bootstrap method in the case of comparing model performance because of the low variance in the performance estimate.

For larger sample sizes, the authors again recommend a 10-fold cross-validation approach in general.

Validation and Test Sets Disappear

It's likely the 3 sets of train, validation, & test sets will not be used nowadays in applied machine learning. Instead of "validation sets," we typically tune hyper-parameters using k-fold cross-validation on the training set, which is defined in pseudocode as follows:

```
# split data
data = ...
train, test = split(data)

# tune model hyper-parameters
parameters = ...
k = ...
for params in parameters:
        performances = list()
        for i in k:
```

```
                fold_train , fold_val = cv_split(i, k, train)
                model = fit(fold_train , params)
                performance_estima te = evaluate(model, fold_val)
                performances.append(performance_estimate)
        performance = summarize(performances)

# evaluate the final model for comparison with other models
model = fit(train)
skill = evaluate(model, test)
```

References to the "test dataset" may also disappear. That is if the cross-validation of model hyper-parameters using the training dataset is nested within a broader cross-validation of the model.

Ultimately, all you are left with is a sample of data from the domain, which we may rightly continue to refer to as the training set.

Note on Cross Validation: The dataset is often split in two (i.e., train and test). Next, with the test set kept aside, randomly choose $X\%$ of the train set to be used as actual training data and the remaining $(100 - X)\%$ partition used as the validation set, where X is a fixed number(e.g., 80%). The model is then trained and validated iteratively. Just split the training set into multiple partitions, then iterate until all data is used for validation in a leave-one-out manner (i.e., train on k-1 and validate on 1, then train on a different k-1 and validate on 1 left out, repeat this until all samples were used 1x for validation). Cross-validation prevents over-fitting, with k-fold Cross Validation being the most popular cross-validation method.

Summary

We just reviewed concepts of the terms "validation dataset" and "test dataset" and how these concepts can be used correctly when evaluating the performance of trained models.

Specifically:

1. There is clear precedent for what "training set", "validation set", and "test set" refer to when evaluating models.
2. "Validation set" is predominately used to describe the evaluation of models when tuning hyper-parameters and data preparation; "Test set" is predominately used to describe the evaluation of a final tuned model when comparing it to other final models.
3. Notions of "validation set" and "test dataset" may disappear when adopting alternate re-sampling methods like k-fold cross-validation, especially when the re-sampling methods are nested.

A FEW NOTES

The remainder of this write-up is the beginning to the remainder of the assignment (i.e., intro for what's still to come/the regression exercises and analysis via sklearn). Thus, it is worth introducing some of the core topics to become familiar with before applying. Note that all hyperlinks were added for a reason, whether an essential reference, a hand-picked tutorial, or a video. We will cover a broad scope of topics throughout the semester, and a sound understanding of these fundamental topics will surely benefit your learning experience. Please take the time to review all text and corresponding links.

sklearn

sklearn provides a class called 'datasets' to interface with ease and dataset-specific utilities for many popular databases available for public download http://scikit-learn.org/stable/datasets/index.html (User Guide). Many of the datasets have built-in methods for splitting the data according to the convention (i.e., convention per a particular task or type of evaluation). Additionally, sklearn.cross_validation.train_test_split sklearn.model_selection.KFold provide interfaces to prepare data split as desired (check out hyperlinks and review API and example uses of each). Take note of specifics, and even try to implement a splitting scheme (this will come soon.

It is worth doing a tutorial, http://scikit-learn.org/stable/tutorial/basic/tutorial.html. Even if some of the concepts are unfamiliar at the moment (e.g., SVD model), lots of time will be spent using sklean from here onward, so could be worth the time to grow more comfortable with the API and online resources. Thus, review the concepts we have covered. Moving forward, consider where we are on the sklearn package flowchart, http://scikit-learn.org/stable/tutorial/machine_learning_map/index.html.

ALL: Do check out this wonderful tutorial on the usage of these features provided in sklearn, https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6. Most certainly worth checking out a blog on linear regression written by the same author, https://towardsdatascience.com/simple-and-multiple-linear-regression-in-python-c928425168f9. The two should provide further insight on the rationale behind using data-splits, the interface available to allow us to easily implement experimental schemes per desire, along with the accessibility of datasets via sklearn, and, finally, a quick view at how models are structured in the package.
Use sklearn as a sample guide or resource for ideas for utilities to be implemented in your project– Think, take notes, and even implement some code in response to the lesson learned here (i.e., add to project code-base). Do not hesitate to share ideas by asking a question, sharing a thought, or expressing a unique perception.

Python Pickle

Now is as good of a time as any to learn about serialization and deserialization in Python via pickle.
Where the concept is pretty straightforward, and I am sure some of you have heard or even use pickle, I do recommend everyone take the time to at least watch this video (youtube resource, peek at the API (above link), and check out notebook example (link in next paragraph).
Why we are learning this– well, it is a core feature of the Python language. Moreover, and among other reasons, trained classifiers are frequently shared throughout the Machine Learning community, which makes the concept of pickling especially relevant (how else would such an object be shared universally without also implementing and sharing some juiced-up parser function... i.e., what pickle does for you under the hood).