Predictive Modeling for Machine Learning Competitions

Ben Herndon-Miller

Houston R Users Group

February 5th, 2019

What is Machine Learning?

- **Machine Learning**'s objective is to use current data to make predictions on future data or learn underlying patterns
- **Statistical Modeling**'s objective is to formalize relationships between variables in the form of mathematical equations

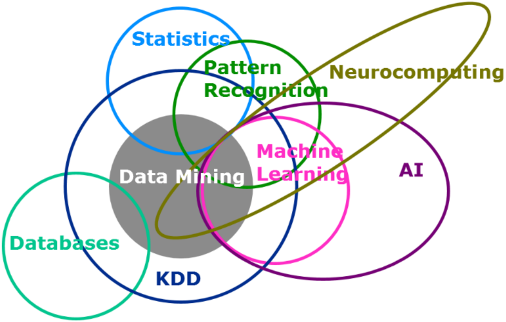


Figure 1: Machine Learning in the Context of the Data Science Universe

Supervised vs. Unsupervised Machine Learning

- **Supervised** Machine Learning's objective is to *predict* labels/outcomes of future data
  - Data structure is a matrix X $(n \times p)$ with a vector Y $(n \times 1)$ of labels/outcomes
  - Each of the n rows of X represents an observation/event/unit of analysis
  - Each of the p columns of X represents a variable or *feature*
  - If $Y \in \mathbb{R}^n$, the task is regression
  - If $Y \in \{0, ..., k\}^n$, the task is classification
- **Unsupervised** Machine Learning's objective is to identify and learn from underlying patterns in data
  - Data structure is a matrix X $(n \times p)$
  - n and p represent the same thing as in supervised learning
  - Common tasks are clustering and pattern recognition

Predict Blood Donations Competition

- **Goal**: predict whether a person donated blood in March 2007
- Data source: UCI Machine Learning Repository
- Data structure:

	Months since Last Donation	Number of Donations	Total Volume Donated (c.c.)	Months since First Donation	Made Donation in March 2007
619	2	50	12500	98	1
664	0	13	3250	28	1
441	1	16	4000	35	1
160	2	20	5000	45	1
358	1	24	6000	77	0

- Submission format:

	Made Donation in March 2007
659	0.5
276	0.5
263	0.5
303	0.5
83	0.5

- Since our objective is prediction, we will use **supervised** machine learning methods

Data Splitting

- In order to predict on unseen (future) data we must split our data into a **training** and **test** set
- **Training** data is used to build or "train" machine learning model
- **Test** data is used to evaluate performance of our trained model

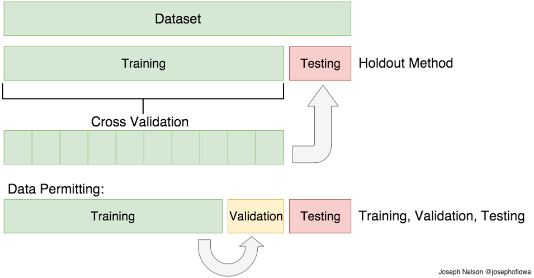


Figure 2: Data Splitting Procedure

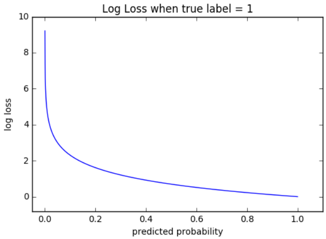
Model Evaluation

- How do we evaluate the predictive ability of a model?
  - Step 1: We train the model on current data of X_{train} and Y_{train}
  - Step 2: We predict the new Y_{test} labels using our trained model on the unseen X_{test}
  - Step 3: We evaluate the predictive accuracy of our model by calculating a function $L(\hat{Y}_{test}, Y_{test})$ called the **loss function**
- A **loss function** is a measurement of error that we seek to *optimize*
- Many loss functions are commonly used:
  - Mean Squared Error for *regression*
  - Misclassification Error for *classification*
  - Log loss (also called Cross Entropy) for *classification*

Log Loss

Log loss is the loss function for the *Predict Blood Donations* competition and has the following equation:

$$Log\ \ loss = -\frac{1}{n}\Sigma_{i=1}^{n}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)]$$



It measures the predictive ability of a classification model that outputs a probability value between 0 and 1. It punishes bad predictions exponentially and rewards models that minimize confidently wrong predictions.

Cross-Validation

How do we optimize our models?

- Many models have what are called "hyper-parameters", which are inputs to the model that are not optimized in the training process (e.g. K in K-Nearest Neighbors)
- **Goal**: choose the optimal values of our hyper-parameters that minimize our loss function
- We accomplish this through a process called *cross-validation*, or repeated data splitting on the training set

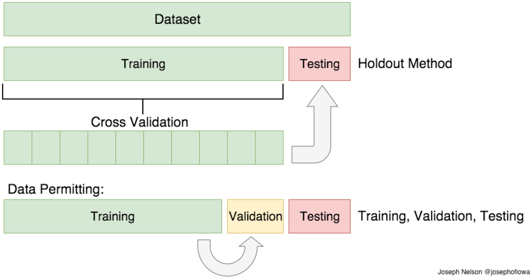


Figure 3: Data Splitting Procedure

Cross-Validation

Remember: in order to evaluate our model without bias we must measure our predictive accuracy on *unseen* data. This is why we use cross-validation.

- There are many methods of cross-validation
  - K-fold cross-validation
  - Repeated cross-validation
  - Leave-one-out cross-validation (LOOCV)
- All of these methods subset the training data into proxy "training" and "test" sets, but during this process they are referred to as "validation" sets
- The concept is the same, we need to predict on *new* data in order to get an unbiased assesment of our model
- Average loss over all iterations of cross-validation to find optimal value of hyper-parameters

Bias-Variance Tradeoff & Overfitting

The bias-variance tradeoff is one of the fundamental theories in machine learning.

- **Bias** is error from poor assumptions of the learning algorithm, high bias results in missing key relationships and *underfitting*
- **Variance** is error from the sensitivity to fluctuations in the the data, high variance can cause an algorithm to model the *noise* and results in *overfitting*

For example, the total Mean-Squared Error is decomposed into Bias, Variance, and random error in the equation below. This tradeoff applies to all forms of supervised learning.

$$E[(Y - \hat{Y})^2] = (Bias[\hat{Y}])^2 + Var[\hat{Y}] + \epsilon$$

Bias-Variance Tradeoff & Overfitting

We can see that when our model is simple, we *underfit* and have **high bias**. In contrast, when we *overfit* our model is too complex and we have **high variance**. The goal of cross-validation is to find the model that minimizes both *bias & variance*.

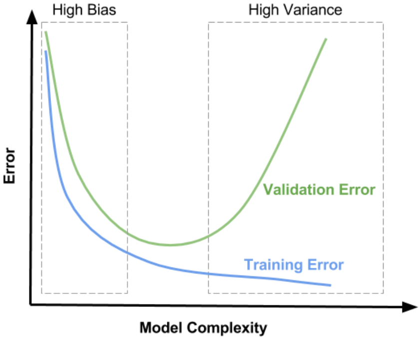


Figure 4: Bias-Variance Tradeoff

Bias-Variance Tradeoff & Overfitting

In the visualization below, the model with low bias and low variance generalizes the underlying relationship of the data far better than either of the first two models. Even though the first model performs well on the training set, it does a poor job generalizing to future data.

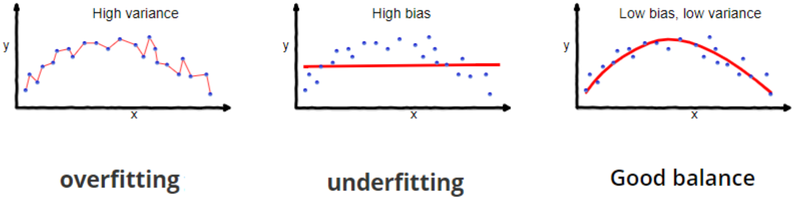


Figure 5: Overfitting and Underfitting

Training the Model

We can avoid overfitting the model if we make sure to always be evaluating our model on new data. Generally, we can accomplish this by following the algorithm below.

```
Define sets of model parameter values to evaluate
for each parameter set do
    for each resampling iteration do
        Hold–out specific samples
        [Optional] Pre–process the data
        Fit the model on the remainder
        Predict the hold–out samples
    end
    Calculate the average performance across hold–out predictions
end
Determine the optimal parameter set
Fit the final model to all the training data using the optimal parameter set
```

Figure 6: Model Training and Parameter Tuning Algorithm

Example: KNN

How would we approach a machine learning problem with the K-Nearest Neighbors algorithm? We use repeated cross-validation in order to find the optimal value of K.

- Split data intro training and testing sets
- For i in 1:20
  - For j in 1:10
    - Randomly subset 70% of the data for training and 30% for validation
    - Fit the KNN model on the training data
    - Predict on the validation set and calculate log loss
  - Calculate average log loss across all iterations of repeated cross-validation
- Select value of K where average log loss from cross-validation is minimized
- Re-train KNN algorithm on entire training set at $K = i_{optimal}$
- Report final prediction error on the test set

References

- Figure 1: https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/
- Figure 2: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6
- Figure 3: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html
- Figure 4: http://www.luigifreda.com/2017/03/22/bias-variance-tradeoff/
- Figure 5: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229
- Figure 6: https://topepo.github.io/caret/model-training-and-tuning.html