ML Basics

Jim Harner

3/31/2018

1.4 Machine Learning Basics

An *algorithm* is a procedure specifying a set of steps to accomplish a task.

Efficient algorithms that work sequentially or in parallel are the basis of pipelines to prepare and process data. The principal data science algorithm types are:

- Data munging, preparation and processing algorithms, such as sorting or MapReduce (data engineering);
- Optimization algorithms for parameter estimation, including stochastic gradient descent, Newton's method, and least squares;
- Machine learning algorithms.

1.4.1 Machine Learning Algorithms

Machine learning algorithms are largely used to predict, classify, or cluster.

- *Prediction* and *classification* are examples of *supervised learning*, whereas
- *clustering* is an example of *unsupervise learning*.

Put another way, supervised learning is concerned with problems that have a measurable (or labeled) response variable and unsupervised learning is concerned with problems without a response variable.

- *Statistical modeling* is done (primarily) to infer the underlying generative process.
- *Machine learning algorithms* are used to predict or classify with the most accuracy. They form the basis of data products.

Consider:

- the interpretation of parameters (real-world interpretation or hand-tuning for predictive power);
- confidence intervals (understanding variability by statisticians vs. prediction by machine learners);
- the role of explicit assumptions (statistical models make explicit assumptions about the generative processes, but the assumptions may be semi-parametric or nonparametric).

The culture wars:

- Statistician: investigates uncertainty and is concerned about sampling validity, but is never certain about anything;
- Data engineer: builds models that predict well with little concern for uncertainty or sampling validity;
- Data scientist: values both approaches.

Data scientist (noun): Person who is better at statistics than any software engineer and better at software engineering than any statistician. (by Josh Wills)

The supervised learning methods are primarily concerned with models of the form:

$$Y = f(X_1, X_2, \ldots, X_p) + \epsilon,$$

where f is an unknown fixed function, Y is the output variable, $X_1, X_2, \ldots, X_p$ are the input variables, and ϵ is the error term which is independent of the X's and has a mean of 0. The output variable is also called the response or dependent variable. The input variables are also called the predictors, features, or independent variables.

The simplest manifestation of this form is the linear regression model given by:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon.$$

Both the input and output variables can be either quantitative (numerical) or qualitative (categorical). The response type restrict the types of models which can be considered. For example,

- if Y is numerical, then we are dealing with a regression model, which can be used for prediction, and
- if it is binary (or more generally categorical), then we are dealing with a logistic regression (or more generally a multinomial model), which can be used for classification.

The types of the predictors are less important since proper encoding can be done for either type of problem.

We use the (training) data to estimate f for either:

- prediction, or
- inference

Prediction

Predictions are given by the estimated model:

$$\hat{Y} = \hat{f}(X),$$

where $\hat{f}$ is the estimated function, $\hat{Y}$ is the predicted output variable, and $X^t = (X_1, X_2, \ldots, X_p)$ is the set of input variables. For prediction the emphasis is on the accuracy of the $\hat{Y}$'s.

Prediction accuracy within a regression context can be defined by:

$$E(Y - \hat{Y})^2 = \left[f(X) - \hat{f}(X)\right]^2 + \ \mathrm{Var}(\epsilon),$$

which partitions accuracy it into two expressions (assuming the X's are fixed). We have no control over $\mathrm{Var}(\epsilon)$ once the data is collected (this is a statistical design issue), but we can reduce the first expression on the right by getting an improved estimator of f. However, as you will see, we need to be careful in how we construct this estimator since we are really interested in predicting the test data. Accuracy within a classification context will be defined when this method is discussed.

Inference

Within a scientific context, it is often of more interest to understand the relationship between the predictors and the response, including which predictors are associated with the response, than in just the accuracy of the prediction model. As such, the interpretation of a model often trumps slight improvements in the model.

It is possible to define measures of flexibility and interpretability for models. Generally, a more flexible model is less interpretable and vice versa. For example, linear regression is relatively interpretable, but has low flexibility. On the other hand, bagging and boosting have limited interpretability, but are very flexible.

Measuring model accuracy

We defined accuracy theoretically in terms of an expected value. Within a regression context, this can be defined in terms of the *mean squared error* defined by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i))^2,$$

where $\hat{f}(x_i)$ is the predicted value at $x_i^t = (x_{i1}, x_{i2}, \ldots, x_{ip})$ for the n observations in the training data set.

More importantly, we need to assess the accuracy of the model in terms of a test data set which was not used to build the model. The mean squared error is defined the same:

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{f}(x_i))^2,$$

for the m observations in the test data set. If these mean squared error are about the same, then overfitting is probably not an issue.

For the classification problem, the most natural measure of accuracy is the classification error rate given by:

$$\frac{1}{n}\sum I(y_i \neq \hat{y}_i)$$

where I is an indicator function which is 1 when $y_i \neq \hat{y}_i$ and 0 otherwise for observations in the training data set. A similar measure of accuracy can be computed for the test data set. This latter measure is an estimate of the true probability of misclassification.

Now consider what happens if the MSE is plotted against a measure of flexibility.

- The MSE for the training dataset will continue to decrease as the flexibility increases. This is a result of fitting the irregularities in the data as the flexibility (degrees of freedom) increases. However, overfitting the training data will not translate to a good fit for the new data.

- The MSE for the test data will initially decrease and then increase as the flexibility increases, i.e., the curve is U shaped. The reason is that flexible models begin fitting noise and training-specific patterns which are not present in the test data.

The variance-bias trade-off

The expected test MSE at $Y = y$ and $X = x$ in the test set is given by:

$$E(y - \hat{f}(x))^2 = \text{Var}(\hat{f}(x)) + \left[\text{Bias}(\hat{f}(x))\right]^2 + \text{Var}(\epsilon)$$

with repeated realizations of training data sets used to estimate f. This expected test MSE is then averaged over all (x, y) in the test set.

Models with high flexibility, e.g., high degrees of freedom, tend to have high variance since changing the training data can result in high variation in $\hat{f}$, whereas models with low flexibility will tend to have low variance since the model is more robust to small changes in the training data. On the other hand, bias results from approximating a complicated real-life model by a simpler one.

Thus, as the flexibility increases, the variance tends to increase and the bias tends to decrease. Since the $\text{Var}(\epsilon)$ is constant, there should be an optimal level of flexibility. We say that:

- models with too much flexibility *overfit* the model, whereas

- models with too little flexibility *underfit* the model.

The optimal flexibility can be found, at least approximately, from the test data or more likely by cross validation.

Cross-validation

Often separate test data is not available. The idea of cross-validation is to hold out a subset of the training observations from the fitting process and then apply the learned model to the held out observations.

Cross-validation can be done in several ways:

- Validation set or held-out set: randomly divide the available data into two parts—the training set and the validation (or test) set. Random splitting can be done multiple times to see the variability in the MSE and how it depends on flexibility. Unfortunately, the validation test error rate can be highly variable.
- Leave-one out: each observation is held out while the remaining observations are used to fit the model. As a result a MSE is computed for each observation. The leave-one-out cross validation (LOOCV) is the average of the individual MSE given by:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \text{MSE}_i,$$

  where $\text{MSE}_i = (y_i - \hat{y}_i)^2$ is the nearly unbiased test error for the i^{th} observation left out based on the fit for the remaining $n-1$ observations. Compared to the validation set approach, the LOOCV has less bias and the data is not randomly divided.
- k-fold: randomly divide the data into k groups and hold out each group successively and use the remaining for the training data. Compute the k-fold estimate of the test data for each group and average:

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} \text{MSE}_i,$$

  where MSE_i is the estimated test error for the left-out group i.

Relative to LOOCV, k-fold CV has more bias, but less variance. For big data, LOOCV is not practical. Often 10-fold cross-validation is used.