Machine Learning, Spring 2019
EN 2105
MWF 9-9:50am

Chao Lan
EN 4084A
clan@uwyo.edu

Machine Learning in Practice

Given a data set of student profiles, how to practically build a student GPA prediction model? Here is a practical pipeline: data preprocessing – model construction – model selection – model training – model evaluation.

Data Preprocessing

We can always examine the data first and properly preprocess it for better model development. There are many things to examine, e.g., missing data, data type and data size.

Missing values are fairly common, but most machine learning algorithms cannot run with them. Statisticians have carefully studied this problem under two scenarios: values are missing at random (MAR) or missing not at random (MNAR) [**?**]. Most machine learning studies just assumes MAR. A naive solution is to remove examples or features with missing values, but this may significantly reduce our data. Another solution is to impute missing values e.g., using statistical imputation methods – a common one is mean imputation, which imputes a missing value by the mean of the corresponding observed values. Let x_{ij} be the j_{th} feature of the i_{th} student. Suppose x_{ij} is missing, and let Ω_j be the index set of students whose j_{th} feature is observed, then mean imputation will fill x_{ij} with[1]

$$\hat{x}_{ij} = \frac{1}{|\Omega_j|} \sum_{k \in \Omega_j} x_{kj}. \tag{1}$$

A third solution is low-rank matrix factorization [**?**], which assumes an incomplete data matrix has low rank and recovers its missing values by first factorizing the matrix and then multiplying back the factors. (We will discuss this later.)

Some features are categorical (e.g., feature 'gender' takes value from {male, female}), but most machine learning algorithms can only deal with numerical features. A naive solution is to encode different categories into different values (e.g., female=1 and male=2), but this introduces additional ordinal information which could mislead learning (e.g., male > female). A common solution is to encode categories using dummy variables, which assigns each of the c categories with a unique c-bit binary string (e.g., female = 01 and male = 10).[2] Note this solution may increase feature dimension e.g., ZIP code feature can be encoded into up to 42k bits.

Some models perform better when features are normally distributed or have similar scales. But if raw features do not have these properties, we can standarize or normalize them. Standardization makes the feature distribution more normal. We standarize x_{ij} by

$$\tilde{x}_{ij} = \frac{x_{ij} - m_j}{\sigma_j}, \tag{2}$$

[1] Sometimes we may also impute along columns instead of rows, depending on the context.
[2] Some also use (c-1)-bit strings.

where m_j and σ_j are the mean and variance of feature j respectively (which can be estimated from data). Normalization makes different features have similar scales. There are many normalization methods, and one is to normalize x_{ij} by

$$\tilde{x}_{ij} = \frac{x_{ij} - \min(x_{\cdot j})}{\max(x_{\cdot j}) - \min(x_{\cdot j})}. \tag{3}$$

But note that neither standardization nor normalization guarantees performance improvement.

We can also examine data size and decide if the sample/feature size is too big/small. If sample size is too big (e.g., 100 million students), we may think about computation and memory efficiency – do we need HPC or parallel/distributed computing (e.g. MapReduce)? can we remove some examples? or, perhaps more fundamentally, do we need all examples to build an accurate model ? do we have to load all data into memory for model training, or can we just stream them (e.g., using online learning algorithms)? If feature size is too big, we may think about reducing it (e.g., using feature selection or feature transformation methods).

There are other things to examine e.g., do we have sequential data, relational data or attribute-based data? do we have data batch or data stream? is our data linear separately?

Model Construction

A machine learning model is a mathematical function with unknown parameters (to be learned from data). Model construction is the process of designing the function and unknown parameters. A big part of this course is to study the designs of classic machine learning models.

We don't have to design new models, though. Many machine learning models are matured and well implemented (e.g., in Python libraries). In practice, we can just try them and pick up the best for our task. If the selected model can meet our performance need, then that's perhaps all we need to know about machine learning – we don't need this course. But what if it can't, or can it do better, or what if it isn't (directly) applicable to our problem? In these cases, it can be helpful to have deeper understandings on these models and the capability of manipulating them or designing new models for new problems – this is what we may learn from this course.

Model Selection

A model may have hyper-parameters. Model selection is the process of optimizing them (from a pool of candidates). An example method is K-fold cross-validation.

Model Training

Model training is the process of estimating unknown model parameters from data (with *fixed* hyper-parameters). It includes designing an objective function (which often consists of a loss term and a regularization term) and designing an optimizer (e.g., an analytic solution or a gradient descent method).

Model Evaluation

After a model is trained, we can evaluate its generalization performance on *unseen* testing data. This process is model evaluation.

Now we discuss some specific steps one can follow to build a model from a data set.

Step 1 (preferred). Before analysis, we can put data in a matrix form, where each row represents an example (student) and each column represents a feature (e.g. gender); the label (GPA) can be put in the last column. We can store the data matrix e.g., in a csv file.

Step 2. Randomly partition examples into a training set and a testing set. A typical way of partition is 75% examples for training and the rest 25% for testing. Keep in mind that testing set will be hidden throughout the learning process, and will only be used in model evaluation.

Step 3 (optional). Preprocess training data. (data preprocessing)

Step 4. Pick up a model e.g., logistic regression. (model construction).

Step 5. Optimize model hyper-parameters using the *training set*. (model selection)

Step 6. Fix hyper-parameters. Learn model parameters from the training set. (model training)

Step 7 (optional). Preprocess testing data based on step 3.

Step 8. Apply model on the testing set and evaluate its performance. (model evaluation)

Step 9 (preferred). Repeat steps 2-8 over multiple trials and average testing performance. A variant is to modify step 2 so we divide data into K folds, and use K-1 folds for training and the rest for testing; then repeat steps 2-8 until all folds are used for testing.