Handout 01: Statistical Learning (Optional)

Introduction

The general course this year will focus on the application of statistical learning to the analysis of textual corpora. I do understand, however, that some students may be looking for a bit more technical details than we have time to go through in the notes. To address this, I will include several handouts from a previous semester's offering of the class. But please note that these are entirely optional!

Formalisms and terminology

Here we give a formalization of the central elements of supervised learning. The resulting language and terminology will be useful as a reference.

Assume that there exists some unknown function f that maps elements from a set Ω into a set $\mathcal{Y}$,

$$f : \Omega \rightarrow \mathcal{Y}, \tag{1.1}$$

and consider observing tuples, known as *training data*,

$$\left\{\omega_i, y_i = f(\omega_i)\right\}_i, \quad \omega_i \in \Omega,\, y_i \in \mathcal{Y}, \quad i = 1, \ldots n. \tag{1.2}$$

We will avoid writing out a formal definition of the (possibly) random nature of f and the (possibly) random process that generates each ω_i. Doing so here would overly divert from the main discussion; we will define the random nature of the data generation process whenever necessary.

A supervised learning algorithm constructs an estimate $\widehat{f}$ of the function f using the training data. The goal is to minimize errors in estimating f on new data for some loss function $\mathcal{L}$. When predicting a continuous response variable, such as an expected price, common loss functions include the squared or absolute difference between the observed and predicted values,

$$\mathcal{L}(\widehat{f}(\omega_{new}), f(\omega_{new})) = \left|\widehat{f}(\omega_{new}) - f(\omega_{new})\right| \tag{1.3}$$

When the set of responses $\mathcal{Y}$ is finite, as it would be when building a spam prediction algorithm, a common choice of $\mathcal{L}$ is to measure the proportion of incorrectly labeled new observations,

$$\mathcal{L}(\widehat{f}(\omega_{new}), f(\omega_{new})) = \begin{cases} 0, & \widehat{f}(\omega_{new}) = f(\omega_{new}) \\ 1, & \text{otherwise} \end{cases} \tag{1.4}$$

Depending on the application, more complex metrics can be used. When evaluating a medical diagnostic algorithm, for instance, it may make more sense to weight incorrectly missing a serious condition more heavily than incorrectly diagnosing a serious condition.

In nearly all supervised learning tasks, the training data will either be given as, or coercible to, a vector of real values. In other words, we can write the set Ω in Equation 1.1 as $\mathbb{R}^p$ for some number p. Similarly, the prediction task can usually be re-written such that $\mathcal{Y}$ is equal to $\mathbb{R}$—in the case of a discrete set, this can be done by associating each category with an integer-based index. Then, by stacking the n training inputs together, we have

$$X = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_n \end{pmatrix} \in \mathbb{R}^{n \times p}, \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n, \quad y = f(X). \tag{1.5}$$

The matrix X is known as the *feature matrix*. This simplification allows for us to draw on techniques from numerical analysis, functional analysis, and statistics in the pursuit of predictive models. A central theme of this course will be building and evaluating supervised learning algorithms motivated by the study of properties of the matrix X and assumptions made regarding the function f.

If the success of a supervised learning algorithm is defined on data that is, by definition, unavailable, how will it be possible to determine how well a predictive model is able to estimate the function f? One approach is to take the observed tuples of data available for building predictive models and partition them into two subsets. Only one of these partitions is used as the *training set* to produce the estimate $\widehat{f}$. The remaining partition, the *testing set*, is used to evaluate how well the estimate can make predictions on new data. More complex schemes operate similarly by splitting the data multiple times (e.g., *cross-validation*) or include a third partition to allow for tuning hyperparameters in the model estimation algorithm.

When considering the construction of a predictive model, a key concern is the desired capacity, or complexity, of a model building algorithm. A formal definition of a training algorithm's complexity is given by its Vapnik–Chervonenkis (VC) dimension [1], though an informal understanding of complexity will suffice here. A model that is overly complex will *overfit* to the training data; that is, $\widehat{f}$ will fit the training data very closely but not be able to generalize well to the testing data. Conversely, a model with low complexity may be too constrained to provide a good approximation to f. In a probabilistic framework, this can be seen as equivalent to a trade-off between *bias* and *variance*. A model building algorithm with high complexity will have a large variance (it may overfit, and is therefore highly sensitive to the training data); if it has a complexity that is too low to approximate f, it will provide systematically biased results for some inputs. The study and control of model complexity, in both its numerical and probabilistic forms, is a guiding theme throughout this text.

References

[1] VAPNIK, V., AND CHERVONENKIS, A. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications 16*, 2 (1971), 264.