Lectures on Machine Learning (Fall 2017)

Hyeong In Choi Seoul National University

Lecture 6: Model Selection (Draft: version 0.8.1)

Topics to be covered:

- Underfitting and overfitting phenomenon
- Regularization: ridge, lasso, elastic net
- Model and algorithm
- Testing, validation and model selection

1 Underfitting and overfitting phenomenon

Figure 1 shows four cases of regression by polynomials of degree 6, 11, 16 and 21, respectively. As we can easily discern, the regression by degree 6 polynomial is not expressive enough to cope with the complex pattern of the data, meaning a higher degree polynomial is needed to do the job properly. This kind of phenomenon is called **underfitting**. On the other hand, the regression by degree 21 polynomial is a perfect fit for all the data points (i.e. the graph of this polynomial goes through all the data points). However, this polynomial swings too. It seems that degree 21 gives too much freedom. This kind of phenomenon is **overfitting**. The regressions by polynomials of degree 11 and 16 seem to hit the data points pretty much the same, but the degree 16 regression seems to swing somewhat more wildly than the degree 11 regression.

These phenomena are associated with the empirical and generalization errors discussed in Lecture 1. In short, underfitting means high empirical error, and its cure is simple. Namely, we adopt a more sophisticated or complex regression function, or model. (For the formal definition of model, see Section 2.) On the other hand, the overfitting phenomenon usually occurs when too complex a model is used, and this is the usual problem one encounters in machine learning.

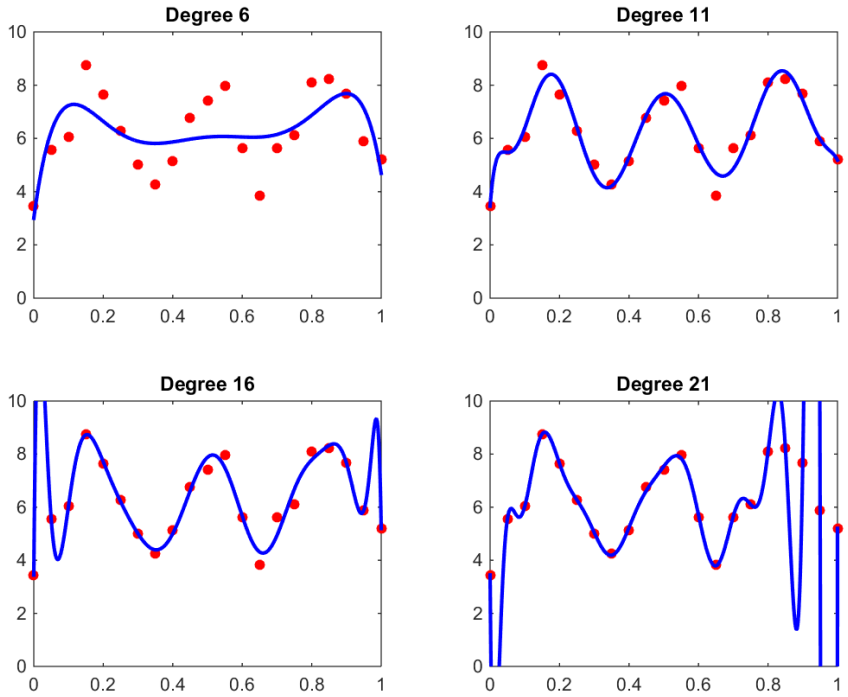

Figure 1: Examples of underfitting & overfitting

2 Regularization: ridge, lasso, elastic net

The overfitting phenomenon we see in Figure 1 occurs because some of the coefficients of the polynomial are too big. As we have seen in Lecture 5, the polynomial regression is in fact a linear regression using monomials as

basis functions. So the real issue is how to control the size of the parameters of linear regression.

A typical way of controlling the parameter of linear regression is the so-called **ridge regression**. Recall that in Lecture 5, the error in multi-dimensional linear regression is of the form

$$E = \frac{1}{2}|X\theta - Y|^2.$$

For ridge regression, we change it to the following:

$$\begin{aligned} E &= \frac{1}{2}|X\theta - Y|^2 + \frac{\lambda}{2}|\theta|^2 \\ &= \frac{1}{2}(X\theta - Y)^T(X\theta - Y) + \frac{\lambda}{2}\theta^T\theta \\ &= \frac{1}{2}(\theta^T X^T X\theta - Y^T X\theta - \theta^T X^T Y + Y^T Y) + \frac{\lambda}{2}\theta^T\theta. \end{aligned}$$

Take the derivative with respect to θ to get the following:

$$\frac{\partial E}{\partial \theta} = X^T X\theta - X^T Y + \lambda\theta.$$

Setting it equal to zero and solving for θ, we get the following solution:

$$\theta = (X^T X + \lambda I)^{-1} X^T Y.$$

Note that $X^T X + \lambda I$ is invertible for any positive λ, because $X^T X$ is positive semi-definite. Note also that if λ is big, the term $\lambda|\theta|^2$ becomes more important. Hence the optimizer will try more to make this term smaller, which results in smaller $|\theta|$. if λ is small, then the optimizer will concentrate its effort more to the empirical error term $|X\theta - Y|^2$, which may result in bigger θ.

A slightly more general version of ridge regression is the **Tikhonov regularization** which uses the error function

$$|X\theta - Y|^2 + |\Gamma\theta|^2,$$

where Γ is a suitable non-singular matrix, called the Tikhonov matrix.

If one uses an error of the form

$$|X\theta - Y|^2 + \lambda|\theta|,$$

the resulting regression is called a **lasso regression**. This simple innocuous looking change has drastically different consequence in that the resulting parameter set tends to be sparse, meaning the number of parameters with zero value increases. When the error is of the form

$$|X\theta - Y|^2 + \lambda|\theta|^2 + \mu|\theta|,$$

the resulting regression is called the elastic net. It combines the advantages of the ridge and lasso regressions. For more details on lasso and elastic net, the reader is referred to the book by Hastie, Tibshirani and Friedman [1].

Remark. *It should be noted that these regularization ideas are not confined to regression.* They can be equally well applied to classification problems like logistic regression. *The same idea and the same techniques are is extensively used in neural networks as well.*

3 Model and algorithm

Figure 1 is a good example of the importance of choosing the *right* kind of polynomial. The question is how would one know which one is right in the first place. The same question applies to classification problems.

In general, in machine learning one is confronted with so many choices of models. Take the binary classification problem, for example. There is a bewilderment of possibilities. One can use logistic regression or other methods like SVM or random forest (to be covered later). One can also use neural networks. Even if one decides to use one method, say logistic regression, there still remains a host of further questions. For instance, should one use linear classification boundary, or nonlinear classification boundary via basis functions? Or, in case one uses the kernel method (to be covered later), one has to decide which kernel to choose out of many possibilities.

All these questions pertain to the so-called **model selection** problem. For the rest of this lecture, we address it from a general point of view. Before we begin, it is better to clarify what we mean by *model*. The common sense meaning of model is the representation of a certain aspect of reality. In machine learning, the reality is the data. But by data, one has to take into account not only that given at present but also the *similar* ones to arise in the future. In other words, the purpose of supervised learning is to find a predictor $f : \mathfrak{X} \to \mathfrak{Y}$ so that $f(x^{(i)}) \approx y^{(i)}$ for any data point $(x^{(i)}, y^{(i)})$ given

at present or in the future. The question is what is the proper form of f to begin with. To answer it, we need the following

Definition 1. *A* **hypothesis space** $\mathcal{H}$ *is a set of maps* $f : \mathfrak{X} \to \mathfrak{Y}$*. Namely,*

$$\mathcal{H} = \{f \mid f : \mathfrak{X} \to \mathfrak{Y}\}.$$

For instance, the set $\mathcal{H}_m$ of polynomials of degree m

$$\mathcal{H}_m = \{f \mid f = \theta_0 + \theta_1 x + \cdots + \theta_m x^m\}$$

is an example of a hypothesis space (for polynomial regression).

Remark. *Many people use the term* **model** *to mean more or less the same as hypothesis space. However, strictly speaking, model is a broader concept. It not only specifies the set of predictors but also encompasses various assumptions on the probability model and on the data sampling, like whether the data is sampled in IID manner or whether the data has a certain relation to each other. In practice, however, model and hypothesis space are frequently used interchangeably, and we will also do so for the rest of this course. The reader must discern the difference in context.*

In model or hypothesis space, there are two kinds of parameters and we have to distinguish them:

Definition 2. *The parameters that are to be fixed (learned or trained) from the data are called* **model parameters***, or in short,* **parameters***. The parameter relating to the model but are fixed a priori and remain fixed throughout the learning (training) process is called* **model hyperparameters** *or in short,* **hyperparameters***.*

In the above polynomial model $\mathcal{H}_m$, $\theta_0, \cdots, \theta_m$ are model parameters, while m is a hyperparameter. If we use regularization, the constants λ or μ in the ridge, lasso and elastic net regressions are also hyperparameters. Then the question naturally arises as to how to choose the best hyperparameters. For that, there are several methods, and one of the actively researched ones is called the Bayesian (hyperparameter) optimization. However, we will not get to it in this course.

The above $\mathcal{H}_m$ is a collection of predictors of the same nature (e.g., coming from the same formula, differing only in the values of parameters). However,

there is no a priori reason why it has to be the case. One may mix predictors of different origin, although such practice may complicate the ensuing algorithm.

Once the model is set, the next task is to find a *good* predictor. The process of finding such a good predictor is called an algorithm. The following Figure 2 shows a schematic on how it is done. **Algorithm** is essentially a

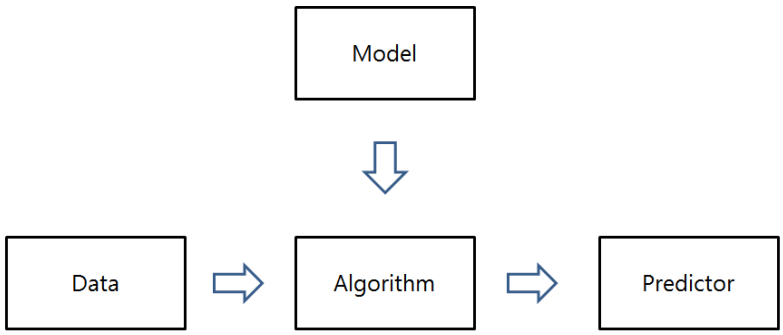

Figure 2: Machine learning framework

procedure of utilizing the data to choose from the hypothesis space of the model the most suitable predictor. While the model can be viewed as a general description of the problem, the algorithm is what makes computer programming work. So to be correct, for a single model, there may be several algorithms that can realize the goal. However, as people confuse model with hypothesis space, people use model and algorithm more or less interchangeably. We will also do so in this course. Again, the reader is advised to understand the meaning of the terms in context.

4 Testing, validation and model selection

A good predictor must satisfy dual criteria. It has to perform well for the given dataset $\mathfrak{D}$, but it also has to work well for *similar* future data. In Figure 1, the data points are marked in red. To illustrate how each of the found regressors behaves in relation to a similar future dataset, we generated new data points and marked them in magenta color in Figure 3. This time

there are twice as many data points compared with the former case. They are still generated by the same formula, except in this case we used a different seed in the random number generator. We can easily observe that the overfit polynomials, say the ones of degree 16 and 21, exhibit big error near the end of the interval where they oscillate wildly, while the degree 11 polynomial shows no such bad behavior. Using the theoretical language introduced in Lecture 1, we say that the *generalization error* is big for the polynomials of degree 16 and 21. In practice, however, we have no luxury of knowing, at

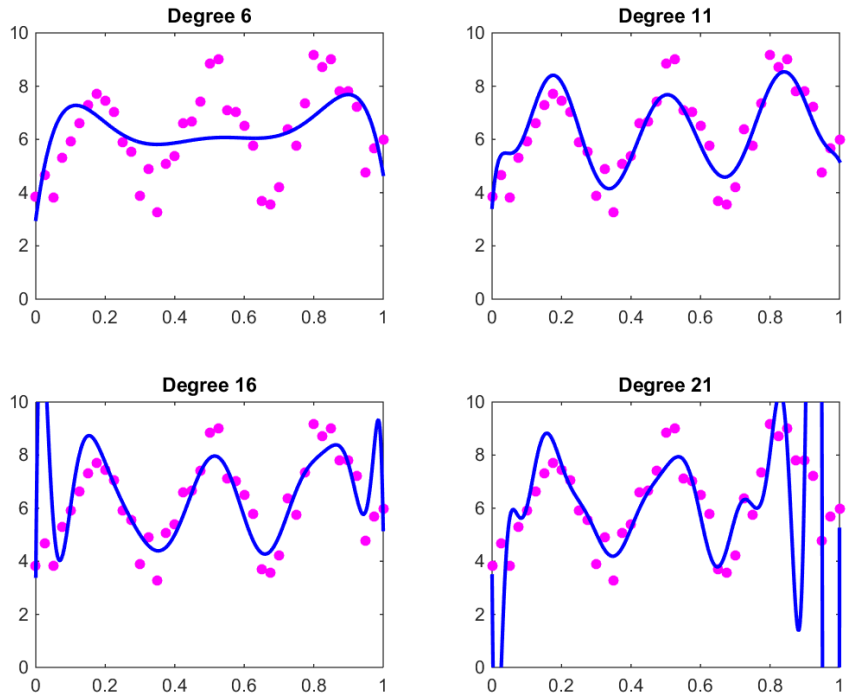

Figure 3: Same polynomials with different random number generator

the time when we create a predictor, how our chosen predictor will perform with *future* dataset. We will only know that our vaunted predictor works pretty badly only at the run-time, i.e. during real operation–a disaster we must avoid. So we somehow must devise a method of guessing or estimating the performance of our predictor with respect to the future dataset.

4.1 Training and testing

One simple way of getting around the difficulty above is to divide the dataset into training set and testing set as depicted in Figure 4. The test set $\mathfrak{D}_{test}$ is held out during the training (learning) so that the predictor is created only with the training set $\mathfrak{D}_{train}$. Once the predictor is found, we use $\mathfrak{D}_{test}$ to test its performance as a proxy for similar future datasets. There is no hard and fast rule for the ratio between these two sets, but it is typical to have a 75% training and 25% test set division. It is also a good practice to randomly put data points in either set to avoid any bias.

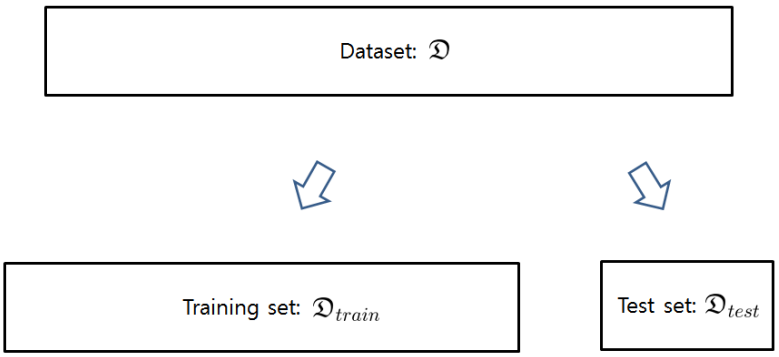

Figure 4: Decomposition into training ans test sets

4.2 Validation and model selection

The above simple scheme of training and testing is good for estimating the performance of a predictor. However, when there are several models to choose from, we need a different scheme. In the case of Figure 1, there are four models: $\mathcal{H}_6$, $\mathcal{H}_{11}$, $\mathcal{H}_{16}$, and $\mathcal{H}_{21}$. We get the *best* predictor from the model, each of which is drawn in Figure 1. Our next task is to decide which one to take as our *final* predictor. This process illustrates the essence of *model selection*. In cases where we have enough data, a typical way is to divide the dataset into three parts, training, validation and testing sets, as in Figure 5.

Suppose there are M models $\mathcal{H}_1, \cdots, \mathcal{H}_M$. The model selection process goes as follows:

(1) Use $\mathfrak{D}_{train}$ to get the *best* predictor $\widehat{\varphi}_a$ from model $\mathcal{H}_a$ for every $a = 1, \cdots, M$.

(2) Compute the error $E_{val}(\widehat{\varphi}_a)$ of $\widehat{\varphi}_a$ with respect to the validation set $\mathfrak{D}_{val}$.

(3) Let $a^* = \underset{a}{\operatorname{argmin}}\, E_{val}(\widehat{\varphi}_a)$. Define the final predictor as $\widehat{f} = \widehat{\varphi}_{a^*}$. Namely the final predictor $\widehat{f}$ is the one with smallest E_{val}.

(4) Use $\mathfrak{D}_{test}$ to compute the error $E_{test}(\widehat{f})$ of $\widehat{f}$.

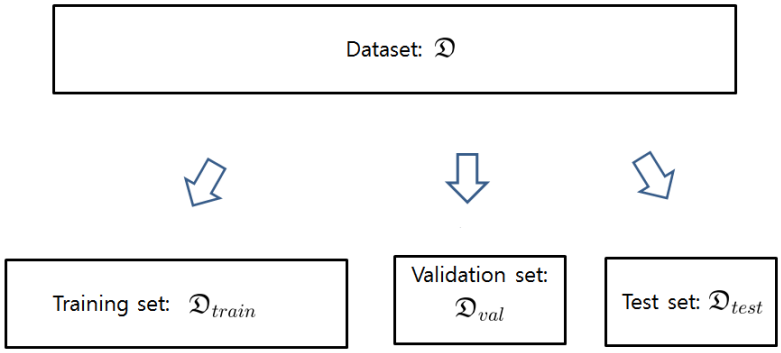

Figure 5: Decomposition into training, validation and test sets

The reason we use $\mathfrak{D}_{val}$ in Stage (2) above is that it was not used in training each predictor $\widehat{\varphi}_a \in \mathcal{H}_a$. Thus the error E_{val}, called the validation error, serves as a proxy for generalization error. That is why it is reasonable to choose the final predictor $\widehat{f}$ as the one with smallest E_{val} in Stage (3). If we use a regularizer like the ridge, lasso or elastic nets, there are two ways to handle it. One is to incorporate each regularizer parameters like λ or μ as part of the model. Typically, one discretizes these parameters and create new models for each discrete parameter value. If so, the above procedure works the same way except that there may be too many models to deal with.

Another way is to find the *best* model $\mathcal{H}_a^*$ by the formula $a^* = \operatorname{argmin}_a E_{val}(\widehat{\varphi}_a)$ without using the regularizer, and then using $\mathfrak{D}_{train} \cup \mathfrak{D}_{val}$ to figure out the *best* regularizer parameter. This method is one of several possible variations along this line. In Stage (4), the error $E_{test}(\widehat{f})$ can now be regarded as a proxy for the generalization error of the final predictor $\widehat{f}$.

As for the right way of dividing the data into three parts, there again is no hard and fast rule. One can use any reasonable ratio, but typically people use a 50:25:25 division ratio.

4.3 k-fold cross validation

The above method of dividing the dataset into three parts is a pretty good one that is also quite popular. However, when we do not have enough data, this method may not be applicable. A popular method to use in this case is the so-called k-fold cross validation. It simply divides the original dataset into equal-sized k subsets $\mathfrak{D}_1, \cdots, \mathfrak{D}_k$ as depicted in Figure 6. We

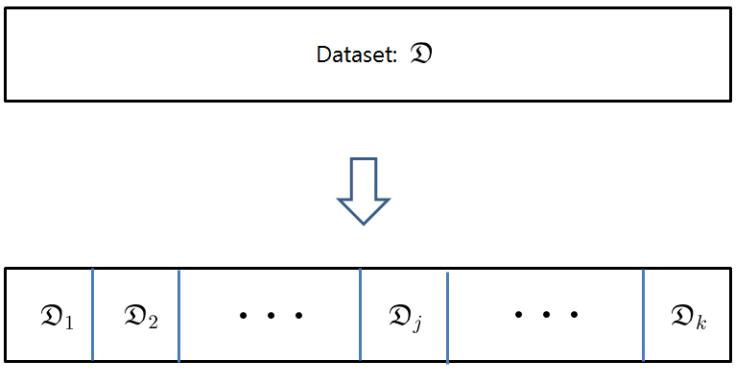

Figure 6: k-fold cross validation set

should note that the manner in which each data point is included in any of these subset has to be random to avoid any bias. For each $j = 1, \cdots, k$, define

$$\mathfrak{D}^{(j)} = \mathfrak{D} - \mathfrak{D}_j.$$

This way the dataset $\mathfrak{D}$ is divided into $\mathfrak{D}^{(j)}$ and $\mathfrak{D}_j$. The advantage of this division is that for each $j = 1, \cdots, k$, we can use $\mathfrak{D}^{(j)}$ as a training set and

$\mathfrak{D}_j$ as a test set. So let $\varphi_a^{(j)}$ be the *best* predictor we get from the hypothesis space $\mathcal{H}_a$ using $\mathfrak{D}^{(j)}$ as a training set. We also compute the error $E_a^{(j)}$ of $\varphi_a^{(j)}$ with respect to the test set $\mathfrak{D}_j$. We then define the error E_a of the model $\mathcal{H}_a$ by

$$E_a = \frac{1}{k} \sum_{j=1}^{k} E_a^{(j)}.$$

This curious quantity serves as a proxy for the generalization error of the *hypothetical best predictor* one can get from model $\mathcal{H}_a$, even if we did not find such a single best predictor and there is no held-out set serving as a test set. We define

$$a^* = \operatorname*{argmin}_a E_a.$$

We then select $\mathcal{H}_{a^*}$ as the *best* model. Once the best model is selected, we use the whole dataset $\mathfrak{D}$ as a training set to get the final predictor $\widehat{f}$ from $\mathcal{H}_{a^*}$. We then take E_{a^*} as a proxy for the generalization error of $\widehat{f}$.

There is no hard and fast rule for what this k has to be, but using $k = 10$, i.e. the 10-fold cross validation, is pretty popular.

4.4 Leave one out cross validation (LOOCV)

When there is a dearth of data, none of the above division methods is practicable. So we go to extreme N-fold cross validation, where N is the number of data points. Thus in this case, each subdivision $\mathfrak{D}_j$ consists of a single data point, and the resulting cross validation is called the **leave-one-out-cross-validation (LOOCV)**.

Similarly one can leave k data points out, which is called the **leave-k-out-cross-validation**.

References

[1] Hastie, T., Tibshirani, R, and Friedman, J., *The Elements of Statistical Learning, 2nd edition*, Springer-Verlag (2009)