Machine Learning Course - CS-433

Model Selection

Oct 10, 2017

major changes by Rüdiger Urbanke 2016

minor changes by Rüdiger Urbanke 2017

Last updated: October 10, 2017

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

We have seen in ridge regression, that the parameter $\lambda > 0$ can be tuned to reduce overfitting by reducing model complexity,

$$\min_{\mathbf{w}} \quad \frac{1}{2N}\sum_{n=1}^{N}[y_n - \mathbf{x}_n^\top \mathbf{w}]^2 \quad + \quad \frac{\lambda}{2N}\sum_{j=0}^{M} w_j^2$$

The parameter λ is typically referred to as *hyper*parameter. In the same manner, we have seen that we can augment the feature vector $\mathbf{x}$ to enrich the model complexity. E.g., in your exercises you have fitted a given data set to polynomial bases of increasing degrees. Here the degree d is the hyperparameter. Similar situations arise when we talk about neural nets where e.g., the width or depth of the network is the hyperparameter.
In all these cases we are faced with the same problem: how do we choose these hyperparameters? This is the *model selection* problem.

Probabilistic Setup

In order to answer the above question in a meaningful way we need to describe our mathematical framework. A fruitful and common such framework is probabilistic. We assume that there is an underlying distribution, call it $\mathcal{D}$, with range $\mathcal{X} \times \mathcal{Y}$. We do not know this distribution but we assume that the data set we see, call it S, consist of independent samples from $\mathcal{D}$:

$$S = \{(\mathbf{x}_n, y_n) \text{ iid} \sim \mathcal{D}\}_{n=1}^{N}.$$

Based on this sample the *learning algorithm* computes the "best" model within the class of given models. E.g., for ridge regression, with a fixed parameter λ, the learning algorithm computes the best weight function $\mathbf{w}$. We have seen that we can use e.g. (stochastic) gradient descent or least-squares for the ridge-regression model as an efficient way of implementing this learning. Let us write more generically $f_S = \mathcal{A}(S)$, where $\mathcal{A}$ denotes the learning algorithm, which depends on the sample S we are given and f_S is the "prediction function." E.g., in our previous example we have $f_S(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}$, where $\mathbf{w}$ is the weight vector we computed. If we want to indicate that f_S also depends on parameters of the model, e.g., the λ in the ridge regression model, we can add a subscript to write $f_{S,\lambda}$.

Training Error versus Generalization Error

Given a prediction function f, how can we assess if f is any good? If we knew the distribution $\mathcal{D}$, then what we should do is to compute the *expected* error over all samples chosen according to $\mathcal{D}$, i.e., we should compute

$$L_{\mathcal{D}}(f) = \mathbb{E}_{\mathcal{D}}[\ell(y, f(\mathbf{x}))],$$

where $\ell(\cdot, \cdot)$ is our loss function. E.g., we have

$$\ell(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2 + \lambda \|\mathbf{w}\|^2,$$

for the the ridge regression problem. This quantity has many names: generalization error, (true/expected) risk, (true/expected) loss, to name just the most common ones. This is the quantity we are fundamentally interested in, but we cannot compute it since $\mathcal{D}$ is not known.
We have given a sample S. It is therefore natural to compute the equivalent *empirical* quantity

$$L_S(f) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f(\mathbf{x}_n)). \tag{1}$$

One of the problems with (1) is that in applications the prediction function f is *itself* a function of the data S. So in fact, we compute the quantity

$$L_S(f_S) = \frac{1}{|S|} \sum_{(\mathbf{x}_n, y_n) \in S} \ell(y_n, f_S(\mathbf{x}_n)).$$

This is often called the *training error* and we have already discussed in previous lectures that this training error might not be representative of the error we see on "fresh" samples. The reason that $L_S(f_S)$ might not be close to $L_{\mathcal{D}}(f_S)$ is of course overfitting.

Splitting the data

To avoid that we validate our model on the same sample we trained it on, it is natural that we split the data into a *training* and a *validation* set, e.g. 80% as training data and 20% as validation data, call them S_t and S_v, respectively. We apply the learning algorithm $\mathcal{A}$ to the training set S_t

and compute the function f_{S_t}. We then compute the error on the validation set, i.e., we compute

$$L_{S_v}(f_{S_t}) = \frac{1}{|S_v|} \sum_{(y_n, \mathbf{x}_n) \in S_v} \ell(y_n, f_{S_t}(\mathbf{x}_n)).$$

Since S_v is a "fresh" sample we can hope that $L_{S_v}(f_{S_t})$ is close to the quantity $L_{\mathcal{D}}(f_{S_t})$. Indeed, in *expectation* both are the same, i.e.,

$$L_{\mathcal{D}}(f_{S_t}) = \mathbb{E}_{S_v \sim \mathcal{D}}[L_{S_v}(f_{S_t})], \tag{2}$$

where the expectation is over the samples of the validation set. Of course, for a particular validation set S_v they might differ (e.g., there is fluctuation in $L_{S_v}(f_{S_t})$ due to the randomness of the validation set).
But we payed a price. We had to split the data and now have less data both for the learning as well as the validation task.
A common trick to avoid this "loss of data" is to use "cross validation" techniques described at the end of this lecture. These techniques work well in practice but are hard to analyze.

Model selection

Let us summarize: We are looking for a way to select (hyper)parameters of our model, like the parameter λ for the ridge regression problem. We split our data into one training set S_t and one validation set S_v and we think of them as having been generated independently and sampled according to the underlying but unknown distribution $\mathcal{D}$. We have in addition a set of values for a parameter of the model, e.g., the parameter λ in the ridge regression problem. Let these values be λ_k, $k = 1, \cdots, K$. To keep things simple we assume that K is some finite value. The basic idea of this bound can be carried over to the case where we have infinitely many models. In this case a more sophisticated concept, called the VC-dimension, is used: as long as we have models with a *finite* VC-dimension then the bound has the same form, with K replaced by the VC dimension.

We run the learning algorithm K times on the same training set S_t to compute the K prediction functions f_{S_t,λ_k}. For each such prediction function we compute the validation error $L_{S_v}(f_{S_t,\lambda_k})$. We then choose that value of the parameter λ which gives us the smallest such validation error.

In the figure below, we plot the validation[1] error (red) as well as the training error (blue) for many values of λ (grid search).

Even if we use separate data sets for training and validation we are faced with several questions.

The first question concerns the learning. How do we know that the function $f_{S_t,\lambda}$ is a good approximation of the function $f_{\mathcal{D},\lambda}$, the best model within our class for the underlying distribution? We will discuss this issue in our next lecture.

The second question concerns the validation: How do we know that, for a fixed function f, $L_{S_v}(f)$ is a good approximation to $L_{\mathcal{D}}(f)$? In other words, how do we know that the empirical error is close to the true error. Let us discuss this

[1] It is called test error in the figure, another common name.

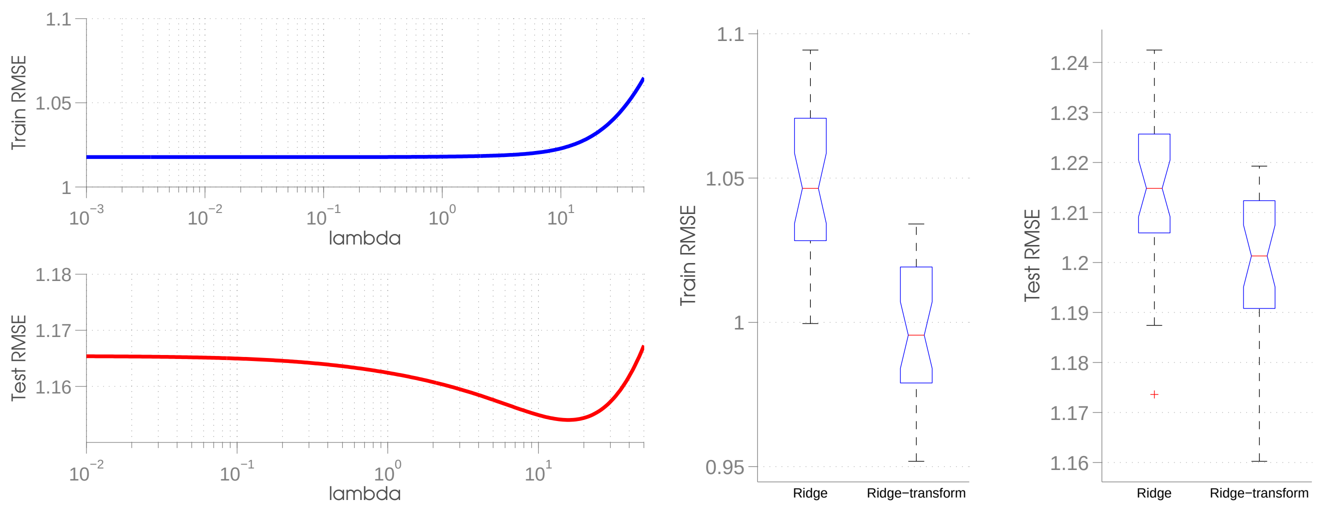


Figure 1: The left figure shows ridge regression results for a 50-50 split. The right one shows a comparison with and without feature transformations. The improvement is very little and might be insignificant.

issue next.

True Error versus Empirical Error

Assume that we have K prediction functions f_k, $k = 1, \cdots, K$, and that our loss function $\ell(\cdot, \cdot)$ is bounded, lets say in $[0, 1]$. We are given a validation set S_v whose samples are chosen iid according to the underlying distribution $\mathcal{D}$. We compute the K empirical errors

$$L_{S_v}(f_k) = \frac{1}{|S_v|} \sum_{(\mathbf{x}_n, y_n) \in S_v} \ell(y_n, f_k(\mathbf{x}_n)).$$

Associated to these are the true errors

$$L_{\mathcal{D}}(f_k) = \mathbb{E}_{(y,\mathbf{x}) \sim \mathcal{D}}[\ell(y, f_k(\mathbf{x}))].$$

How far are these apart? We have seen in (2) that in expectation they are the same. But we need to worry about the variation due to the finite size of the validation set. We claim that

$$\mathbb{P}\left\{\max_k |L_{\mathcal{D}}(f_k)) - L_{S_v}(f_k)| \geq \sqrt{\frac{\ln(2K/\delta)}{2|S_v|}}\right\} \leq \delta. \quad (3)$$

This bound gives us some nice insights. First, the error goes down at least like one over the square root of the number of validation points. The more data points we have therefore, the more confident we can be that the empirical loss we measure is close to the true loss. Further, if we test K hyper-parameters our error only goes up by a very small factor which is proportional to $\sqrt{\ln(K)}$. So we can test quite a few different models without incurring a large penalty.
The proof of this statement is not very difficult. Since we assumed that each data sample $(\mathbf{x}_n, y_n)$ in the validation set S_v is chosen independently, the associated losses $\ell(y_n, f(\mathbf{x}_n))$, given a fixed model f, are also independent and identically distributed random variables, taking values in $[0, 1]$ by assumption. Call each such loss Θ_n. The expected value of $\Theta_n = \ell(y_n, f(\mathbf{x}_n))$ is equal to the true loss

$$L_{\mathcal{D}}(f) = \mathbb{E}[\ell(y_n, f(\mathbf{x}_n)).$$

The empirical loss on the other hand is equal to the average of $|S_v|$ such idd values.
We want to know the chance that the the empirical loss $L_{S_v}(f))$ deviates from its true value by more than a given constant. This is a classical problem adressed in the following lemma.

Lemma 0.1 (Chernoff Bound). *Let $\Theta_1, \cdots, \Theta_N$ be a sequence of iid random variables with mean $\mathbb{E}[\Theta]$ and range $[0, 1]$. Then, for any $\epsilon > 0$,*

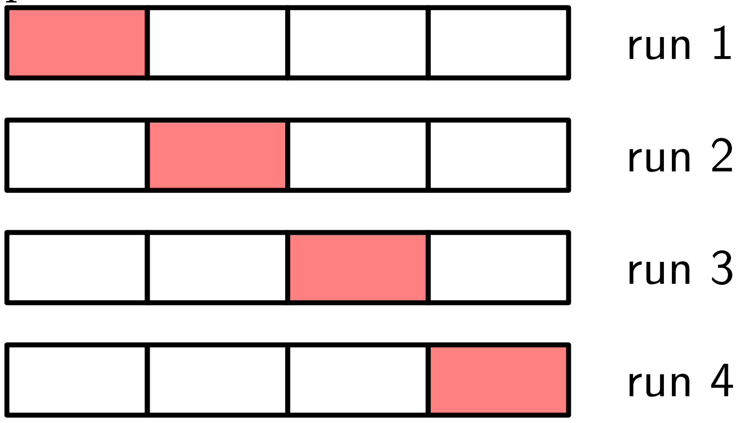
$$\mathbb{P}\{|\frac{1}{N}\sum_{n=1}^{N}\Theta_n - \mathbb{E}[\Theta]| \geq \epsilon\} \leq 2e^{-2N\epsilon^2}.$$

Using Lemma 0.1 let us show (3). Assume first that $K = 1$. Equating $2e^{-2|S_v|\epsilon^2}$ with δ we get that $\epsilon = \sqrt{\frac{\ln(2/\delta)}{2|S_v|}}$ as claimed.
For a general K, if we check the deviations for K independent samples and ask for the probability that for at least one such sample we get a deviation of at least ϵ then by the union bound this probability is at most K times as large as in the case where we are only concerned with a single instance. I.e., the upper bound becomes $2Ke^{-2|S_v|\epsilon^2}$. Hence, equating now $2Ke^{-2|S_v|\epsilon^2}$ with δ we get that $\epsilon = \sqrt{\frac{\ln(2K/\delta)}{2|S_v|}}$ as stated.
For completeness, we state at the very end of these notes a proof of Lemma 0.1.

Cross-validation

Splitting the data once into two parts (one for training and one for testing) is not the most efficient way to use the data. Cross-validation is a popular way of making better use of the data. There are many versions to date. Let us just mention one.
K-fold cross-validation is a popular scheme. Randomly partition the data into K groups. Now train K times. Each

time leave out exactly one of the K groups for testing and use the remaining $K-1$ groups for training. Average the K results. Note that in this way we have used all data for training and all data for testing and we have used each data point the same number of times.



Cross-validation returns an unbiased estimate of the *generalization error* and its variance.

Additional Notes

Proof of Lemma 0.1

Instead of considering the setup in the lemma we can equivalently assume that $\mathbb{E}[\Theta] = 0$ and that the Θ_n take values in $[a, b]$, where $a \leq 0 \leq b$ and $b - a = 1$. We will show that

$$\mathbb{P}\{\frac{1}{N}\sum_{n=1}^{N}\Theta_n xi \geq > \epsilon\} \leq e^{-2N\epsilon^2}.$$

This, together with the equivalent bound

$$\mathbb{P}\{\frac{1}{N}\sum_{n=1}^{N}\Theta_n \leq -\epsilon\} \leq e^{-2N\epsilon^2}$$

will prove the claim. We have

$$\begin{aligned}\mathbb{P}\{\frac{1}{N}\sum_{n=1}^{N}\Theta_n \geq \epsilon\} &\overset{s\geq 0}{=} \mathbb{P}\{e^{s\frac{1}{N}\sum_{n=1}^{N}\Theta_n} \geq e^{s\epsilon}\} \\ &\overset{(a)}{\leq} \min_{s>0}\mathbb{E}[e^{s\frac{1}{N}\sum_{n=1}^{N}\Theta_n}]e^{-s\epsilon} \\ &\overset{(b)}{=} \min_{s>0}\prod_{n=1}^{N}\mathbb{E}[e^{\frac{s\Theta_n}{N}}]e^{-s\epsilon} \\ &= \min_{s>0}\mathbb{E}[e^{\frac{s\Theta}{N}}]^N e^{-s\epsilon} \\ &\overset{(c)}{\leq} \min_{s>0}e^{s^2/(8N)}e^{-s\epsilon} \\ &\overset{s=4N\epsilon}{=} e^{-2N\epsilon^2}.\end{aligned}$$

Here, step (a) follows from the Markov inequality. In step (b) we have used the fact that the random variables Θ_n, and hence the random variables $e^{\frac{s\Theta_n}{N}}$, are independent so that the expectation of the product is equal to the product of the expectations. Finally, in step (c) we have used the so-called Hoeffding lemma. It states that for any random variable X, with $\mathbb{E}[X] = 0$ and $X \in [a, b]$ we have

$$\mathbb{E}[e^{sX}] \leq e^{\frac{1}{8}s^2(b-a)^2}.$$