Lecture 8: Model Selection

CSE 427: Machine Learning

Md Zahidul Hasan

Lecturer, Computer Science
BRAC University

Spring 2023

Non-uniform Learnability

Definition

We say that a hypothesis h is (ϵ, δ)-competitive with another hypothesis h' if the following holds:

$$Pr[R(h) \leq R(h') + \epsilon] \geq 1 - \delta.$$

Definition

A hypothesis class $\mathcal{H}$ is said to be non-uniformly learnable if there exists an algorithm $\mathcal{A}$, and a function $m_{\mathcal{H}} : (0,1)^2 \times \mathcal{H} \to \mathbb{N}$ so that for every $\epsilon, \delta \in (0,1)$ and every $h \in \mathcal{H}$, if we take a sample S with size $m \geq m_{\mathcal{H}}(\epsilon, \delta, h)$, then for every distribution $\mathcal{D}$, we have that, $\mathcal{A}(S)$ is (ϵ, δ)-competitive with h.

In the case of agnostic PAC learning, $R(A(S)) \leq \min_{h' \in \mathcal{H}} R(h')$ $+\epsilon \leq R(h) + \epsilon$ held for a sample size that doesn't depend on hypothesis h. But in non-uniform learning, the sample size depends on the hypothesis h with which $\mathcal{A}(S)$ is competing.

Characterization of Non-uniform Learning

Theorem

A hypothesis class $\mathcal{H}$ of binary classifiers is non-uniformly learnable if and only if it is a countable union of agnostic PAC learnable hypothesis classes.

Theorem

If $\mathcal{H}_1, \mathcal{H}_2, \cdots$ is a countable collection of hypothesis classes where each has the uniform convergence property, then their union is non-uniformly learnable.

Example

Let's say we are trying to solve the problem of binary classification with polynomials. Let's say $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$ where $\mathcal{H}_n$ is the set of all polynomials with degree exactly n. It's easy to prove that $VCdim(\mathcal{H}_n) = n + 1$. So, $VCdim(\mathcal{H}) = \infty$. Using the above theorems, we can say that $\mathcal{H}$ is non-uniformly learnable, but yet it is not agnostic PAC learnable because of it's VC dimension.

Structural Risk Minimization

In the ERM paradigm, we have chosen any hypothesis that gives the minimum empirical error. But notice that,

$$R(h) \leq R_S(h) + \sqrt{\frac{\log|H| + \log\frac{2}{\delta}}{2m}}$$

Therefore, if two hypothesis classes give the same empirical error, the true risk is minimized by a hypothesis class that has a smaller size or a smaller encoding length. Let's say $\mathcal{H}_n$ is the hypothesis class containing all polynomials of degree at most n. Then, $\mathcal{H}_1 \subset \mathcal{H}_2 \subset \mathcal{H}_3 \subset \cdots$. If all of the hypothesis classes give similar empirical error, then we prefer the one with the smallest size. So, we want to be biased towards a particular hypothesis class by giving each hypothesis class a weight so that $\sum_{n=1}^{\infty} w_n \leq 1$. Let's say $\mathcal{H} = \bigcup_{n\in\mathbb{N}} \mathcal{H}_n$ where each $\mathcal{H}_n$ satisfies the uniform convergence property for a sample size called $m_{\mathcal{H}}(\epsilon, \delta)$. Now let's define:

$$\epsilon_n(m, \delta) = \min\{\epsilon \in (0,1) : m_{\mathcal{H}}(\epsilon, \delta) \leq m\}$$

SRM Continued

This is the smallest achievable gap between the empirical error and the generalization error if trained with a sample of size m. Since, uniform convergence holds, we have that, with probability of at least $1-\delta$:

$$\forall h \in \mathcal{H}_n,\ \ |R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \epsilon_n(m,\delta).$$

Lemma

From here, it can be inferred that for all $h \in \mathcal{H}_n$,

$$Pr[|R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \epsilon_n(m, w_n.\delta)] \geq 1 - w_n.\delta \geq 1-\delta$$

We can also prove that, for all $h \in \mathcal{H}$,

$$Pr[|R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \min_{n:h\in\mathcal{H}_n} \epsilon_n(m, w_n.\delta)] \geq 1-\delta$$

Proof: $Pr[|R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \min_{n:h\in\mathcal{H}_n} \epsilon_n(m, w_n.\delta)]$
$= Pr[\bigcap_{n:h\in\mathcal{H}_n} |R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \epsilon_n(m, w_n.\delta)]$
$= 1 - Pr[\bigcup_{n:h\in\mathcal{H}_n} |R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \geq \epsilon_n(m, w_n.\delta)]$

The Structural Risk Minimization Paradigm

$\geq 1 - \sum_{n:h\in\mathcal{H}_n} Pr[|R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \geq \epsilon_n(m, w_n.\delta)]$
$\geq 1 - \sum_{n:h\in\mathcal{H}_n} w_n.\delta \geq 1 - \delta \sum_{n:h\in\mathcal{H}_n} w_n \geq 1 - \delta$. Let's define, $n(h) = \min\{n : h \in \mathcal{H}_n\}$. Since, we have that, $|R_{\mathcal{D}}(h) - R_{\mathcal{S}}(h)| \leq \min_{n:h\in\mathcal{H}_n} \epsilon_n(m, w_n.\delta) \leq \epsilon_{n(h)}(m, w_{n(h)}.\delta)$, we want to find a hypothesis that minimizes $R_S(h) + \epsilon_{n(h)}(m, w_{n(h)}.\delta)$.

Algorithm 1 Structural Risk Minimization

```
Prior knowledge: $\mathcal{H} = \bigcup_n \mathcal{H}_n$ where $\mathcal{H}_n$ has the uniform convergence property.
$w_n \in (0,1) where \sum_n w_n \leq 1$
Input: training set $S \sim \mathcal{D}^m$, confidence parameter $\delta$
Output: $h \in \arg\min_{h\in\mathcal{H}}[R_s(h) + \epsilon_{n(h)}(m, w_{n(h)}.\delta)]$
```

Now, how do we define the weights? In the next slides, we will develop a notion called 'minimum description length' to do that.

Minimum Length Encoding

In this paradigm, we are interested in the hypothesis that has the minimum description length, yet it admits a small empirical error. We will define a description language for the hypotheses that is prefix-free. It means that, the description of one hypothesis can never be a prefix of another hypothesis. Let's say the alphabet of the language is $\Sigma = \{a, b\}$. And $d(\mathcal{H}) = \{aab, ba, bb\}$. Let's say if h is a hypothesis, then $|d(h)|$ is the description length of the hypothesis. We can prove that, $\sum_{h \in \mathcal{H}} \frac{1}{2^{|d(h)|}} \leq 1$. In this example, the maximum length of a description is 3. So, the prefix *ba* covers both *baa* and *bab*. *bb* covers both *bba* and *bbb*. Let's say, if $\mathcal{H}$ included all of $\{aaa, aab, aba, abb, baa, bab, bba, bbb\}$, then $\sum_{h \in \mathcal{H}} \frac{1}{2^3} = 1$. But $\mathcal{H}$ doesn't always include all strings of length 3. And since, $\mathcal{H}$ is a prefix-free set, *ba* and *bb* covers all of the last 4 strings. But *aab* covers only one of the first 4 strings. So, the strings of $d(\mathcal{H})$ contains $k \leq 2^m$ where m is the maximum length of a description. So, we can say that, $\sum_{h \in \mathcal{H}} \frac{1}{2^{|d(h)|}} = \frac{k}{2^m} \leq 1$.

Frame Title

From the above fact, we can give each hypothesis a weight of $w(h) = \frac{1}{2^{|d(h)|}}$. A few lectures ago, we proved that, $R(h) \leq R_S(h) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}$ holds with probability at least $1 - \delta$. So, using the above lemma, we get, $R(h) \leq R_S(h) + \sqrt{\frac{-\log w(h) + \log \frac{2}{\delta}}{2m}}$ holds with probability at least $1 - w(h)\delta \geq 1 - \delta$. If we take $w(h) = \frac{1}{2^{|d(h)|}}$, then we get the following:

$$R(h) \leq R_S(h) + \sqrt{\frac{|d(h)| \ln 2 + \log \frac{\delta}{2}}{2m}} < R_S(h) + \sqrt{\frac{|d(h)| + \log \frac{\delta}{2}}{2m}}$$

Algorithm 2 Minimum Description Length

Prior knowledge: $\mathcal{H} = \bigcup_n \mathcal{H}_n$ where each $\mathcal{H}_n$ has only a single hypothesis.

Input: training set $S \sim \mathcal{D}^m$, confidence parameter δ

Output: $h \in \arg\min_{h \in \mathcal{H}} [R_s(h) + \sqrt{\frac{|d(h)| + \log \frac{\delta}{2}}{2m}}]$

The Discretization Trick

We have a good estimate for the sample complexity necessary to learn finite hypothesis classes and that's:

$$m \leq \frac{\log|\mathcal{H}| + \log\frac{2}{\delta}}{\epsilon^2}$$

But for infinite hypothesis, we can get a glimpse of a practical sample complexity needed to learn infinite hypothesis classes. Let's recall the threshold functions $\mathcal{H} = \{1_{x \leq \theta} : \theta \in \mathbb{R}\}$. This is an infinite hypothesis class consisting of one parameter. But in practice, in most of the programming languages, these parameters are stored using 64 bits. So, in that case, $|\mathcal{H}| \leq 2^{64d}$ where d is the number of parameters necessary to represent the hypothesis class. Using this in the equation above, we get:

$$m \leq \frac{64d + \log\frac{2}{\delta}}{\epsilon^2} = \frac{\left(\frac{64d+\log 2}{\log\frac{1}{\delta}} + 1\right)\log\frac{1}{\delta}}{\epsilon^2} = \frac{g(d)\log\frac{1}{\delta}}{\epsilon^2}$$

where $g(d)$ is an increasing function of d.

Model Selection Using SRM

Let's say $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$ is the set of all polynomial regressors where $\mathcal{H}_n$ is the hypothesis class containing all polynomials of degree at most n. So, in order to express $\mathcal{H}_d$ we need $d + 1$ parameters. Let's give each $\mathcal{H}_d$ a weight of $w_d = \frac{6}{\pi^2 d^2}$. We know that, with probability more than $1 - \delta$, for every $h \in \mathcal{H}_d$:

$$R(h) \leq R_S(h) + \sqrt{\frac{g(d) \log \frac{1}{w_d . \delta}}{m}} = R_S(h) + \sqrt{\frac{g(d)(\log \frac{\pi^2 d^2}{6\delta})}{m}}$$

If we use SRM with the above upper bound, if $h_n \in \mathcal{H}_n$ performs as good as $h_m \in \mathcal{H}_m$ with $n < m$, then SRM is going to choose h_n because it's less complex. But in practice, the bound on the right hand side is unrealistic and pessimistic. So, in the next slide, we will see a more practical approach for selecting a model.

Validation

The upper bounds for the true error that involve the empirical error might be too pessimistic. If we need a more tighter bound and if data is abundant, then we can hold out a portion of the dataset for estimating the risk of a hypothesis. This portion is called the hold-out set or the validation set. The rest of the dataset is called the training dataset. We train our algorithm on the training dataset and test it on the validation dataset to estimate the true error. And this is a better estimate than the empirical error because the learner can be biased towards the training dataset. If the size of our validation data is m_v, then with probability more than $1 - \delta$, we can say using Hoeffding's inequality:

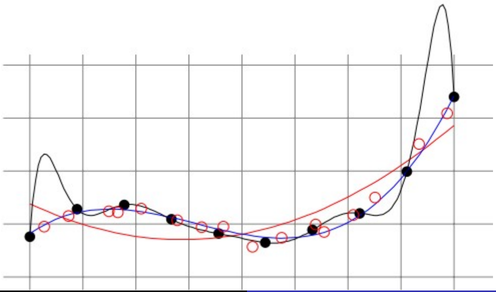
$$R_{\mathcal{D}}(h) \leq R_V(h) + \sqrt{\frac{\log \frac{2}{\delta}}{2m_v}}$$

where $R_V(h)$ is the error in the validation data. Since, m_v is in the range of m, the bound on the right side is tighter than a typical bound involving the empirical risk.

Model Selection Using Validation

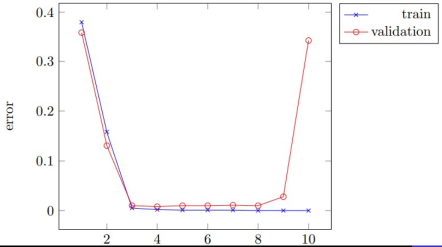
We can run a number of different algorithms on a dataset to learn a hypothesis class $\mathcal{H}$. Let's say that these algorithms output the following hypotheses $\mathcal{H}' = \{h_1, h_2, \cdots, h_f\}$. Then we determine the errors of these hypotheses on the validation dataset. We pick the hypothesis that makes the minimum mistakes on the validation set. We can prove the following using union bound as we did in the case of finite hypothesis classes:

$$Pr[\forall h \in \mathcal{H}', \quad R_{\mathcal{D}}(h) \le R_S(h) + \sqrt{\frac{\log|\mathcal{H}'| + \log\frac{2}{\delta}}{2m_v}}] \ge 1 - \delta.$$



Model Selection Using Validation contd.

In the above image, we can see three polynomials of degree 2, 3 and 10 respectively. The 2 degree polynomial is the red curve. The blue curve has degree 3. And the black curve has degree 10. The dark points indicate the training set. The hollow points indicate the validation set. We can see from the diagram that the polynomial with degree 10 leads to over-fitting. It admits no error on the training set. But even though polynomial of degree 3 admits some errors in the training set, it performs better in the validation set. In the image below, x axis represents polynomial degree and y axis represents error. It's evident that higher degrees lead to overfitting.



K-fold Cross Validation

Most of the time, data is scarce. So, we don't have the luxury of wasting data on validation. But we can divide the dataset into k equal chunks. We can then choose any chunk as the validation data, and train the algorithm on the rest of the chunks. We do this for all of the chunks. And in the end, we take the average of these validation errors to compute the final validation error. K-fold cross validation is often used to tune hyper-parameters. If $k = m$, we call this procedure "leave one out" because the hold-out set consists of just one data instance. In most of the situations, k-fold cross validation works very well. But there are situations where k-fold cross validation might fail.
Usually, we divide our dataset into 3 parts, namely - training data, testing data and validation data. We train on the training data, tune hyper-parameters and select models using the validation data and test our hypothesis on the testing data.

What To Do If Learning Fails

1. Increase data volume
2. Weed out noisy data and features
3. Change hypothesis class by
    - enlarging it
    - reducing it
    - completely changing it
    - changing the parameters
4. change the feature representation of the data
5. change the optimization algorithm