Machine Learning

Model Selection and Validation

Fabio Vandin November 10^{th}, 2023

Model Selection

When we have to solve a machine learning task:

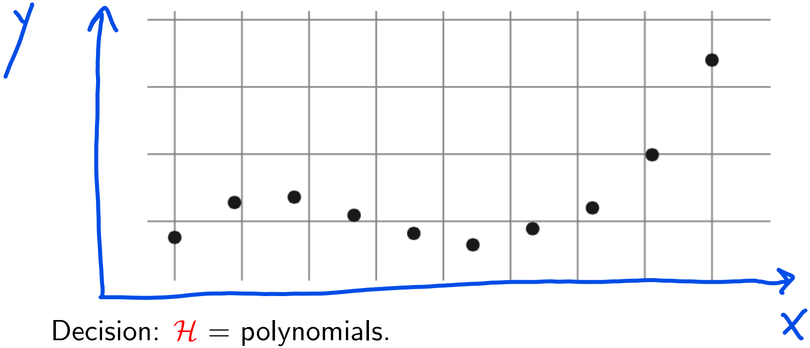
- there are different algorithms/classes
- algorithms have parameters

Question: how do we choose a algorithm or value of the parameters?

Example

$\vec{x} = [x]$

Regression task, $\mathcal{X} = \mathbb{R}$, $Y = \mathbb{R}$



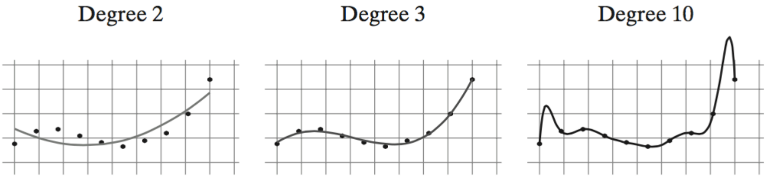
Decision: $\mathcal{H} =$ polynomials.

Note: can be done using the linear regression machinery we have seen!

How do we pick the degree d of the polynomial?

What about considering the empirical risk of best hypothesis of various degrees (e.g., d=2, 3, 10)?

Best hypotheses for degree $d \in \{2, 3, 10\}$



Empirical risk is not enough!

Approach we will consider: validation!

Validation

Idea: once you pick an hypothesis, use new data to estimate its true error

Assume we have picked a predictor h (e.g., by ERM rule on a $\mathcal{H}_d$).

Let $V = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_v}, y_{m_v})$ be a set of m_v *fresh* samples from $\mathcal{D}$ and let $L_V(h) = \frac{1}{m_V} \sum_{i=1}^{m_V} \ell(h, (\mathbf{x}_i, y_i))$

Assume the loss function is in $[0, 1]$. Then by Hoeffding inequality we have the following.

Proposition

For every $\delta \in (0, 1)$, with probability $\geq 1 - \delta$ (over the choice of V) we have

$$|L_V(h) - L_{\mathcal{D}}(h)| \leq \sqrt{\frac{\log(2/\delta)}{2m_v}}$$

Comparison with VC-dimension bound

Assume:

- h has been picked from $\mathcal{H}_d$
- VC-dimension of $\mathcal{H}_d$ is $VCdim(\mathcal{H}_d)$

Then (by fundamental theorem of learning):

$$L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{C\frac{VCdim(\mathcal{H}_d) + \log(1/\delta)}{2m}}$$

where C is a constant.

From previous proposition:

$$L_{\mathcal{D}}(h) \leq L_V(h) + \sqrt{\frac{\log(2/\delta)}{2m_v}}$$

$\Rightarrow$ if we pick $m_v \in \Theta(m)$, the second bound is more accurate!

Note: possible only because we use *fresh* (new) samples...

In practice:

- we have only 1 dataset
- we split it into 2 parts:
  - training set
  - *hold out* or *validation* set

A similar approach can be used for model selection, i.e. to pick one hypothesis (or class of hypothesis, or value of a parameter) among hypothesis in several classes...

Validation for Model Selection

Assume we have $\mathcal{H} = \cup_{i=1}^{r} \mathcal{H}_i$

Given a training set S, let h_i be the hypothesis obtained by ERM rule from $\mathcal{H}_i$ using S
$\Rightarrow$ how do we pick a final hypothesis from $\{h_1, h_2, \ldots, h_r\}$?

Validation set: $V = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{m_v}, y_{m_v})$ be a set of *fresh* m_v samples from $\mathcal{D}$

$\Rightarrow$ choose final hypothesis (or class or value of the parameter) from $\{h_1, h_2, \ldots, h_r\}$ by ERM over validation set

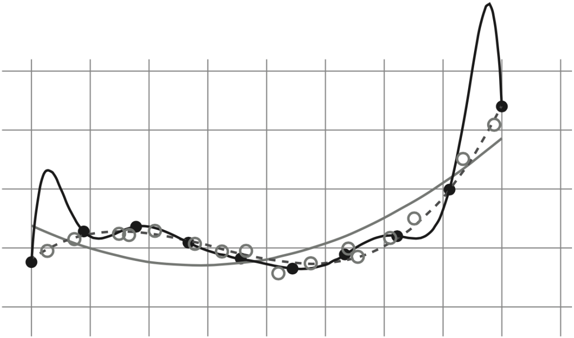
Assume loss function is in $[0, 1]$. Then we have the following.

Proposition

With probability $\geq 1 - \delta$ over the choice of V we have

$$\forall h \in \{h_1, \ldots, h_r\} : |L_{\mathcal{D}}(h) - L_V(h)| \leq \sqrt{\frac{\log(2r/\delta)}{2m_v}}$$

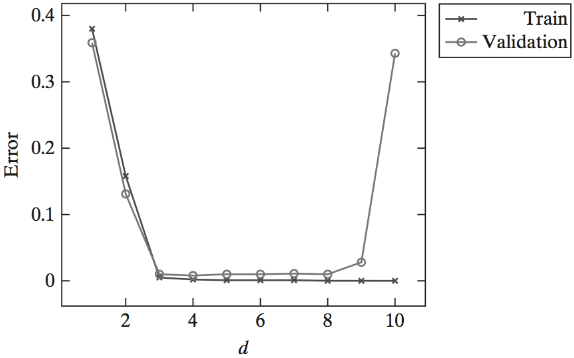
Example



Model-Selection Curve

Shows the training error and validation error as a function of the complexity of the model considered

Example



Training error decreases but validation error increases $\Rightarrow$ *overfitting*

What if we have one or more parameters with values in $\mathbb{R}$?

1. Start with a rough *grid* of values
2. Plot the corresponding model-selection curve
3. Based on the curve, zoom in to the correct *regime*
4. Restart from 1) with a finer grid

Note: the empirical risk on the validation set *is not* an estimate of the true risk, in particular if r is large (i.e., we choose among many models)!

Question: how can we estimate the true risk after model selection?

Train-Validation-Test Split

Assume we have $\mathcal{H} = \cup_{i=1}^{r} \mathcal{H}_i$

Idea: instead of splitting data in 2 parts, divide into 3 parts

1. *training set*: used to learn the best model h_i from each $\mathcal{H}_i$
2. *validation set*: used to pick one hypothesis h from $\{h_1, h_2, \ldots, h_r\}$
3. *test set*: used to estimate the true risk $L_{\mathcal{D}}(h)$

$\Rightarrow$ the estimate from the test set has the guarantees provided by the proposition on estimate of $L_{\mathcal{D}}(h)$ for 1 class

Note:

- the test set *is not involved* in the choice of h
- if after using the test set to estimate $L_{\mathcal{D}}(h)$ we decide to choose another hypothesis (*because we have seen the estimate of $L_{\mathcal{D}}(h)$ from the test set...*)
  $\Rightarrow$ we cannot use the test set again to estimate $L_{\mathcal{D}}(h)$!

k-Fold Cross Validation

When data is not plentiful, we cannot afford to use a *fresh* validation set $\Rightarrow$ cross validation

$\Rightarrow$ k-fold cross validation:

1. partition (training) set into k *folds* of size m/k
2. for each fold:
    - train on union of other folds
    - estimate error (for learned hypothesis) from the fold
3. estimate of the true error = average of the estimated errors above

Lease-one-out cross validation: $k = m$

Often cross validation is used for model selection

- at the end, the final hypothesis is obtained from training on the entire training set

k-Fold Cross Validation for Model Selection

input:

training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
set of parameter values Θ
learning algorithm A
integer k

partition S into $S_1, S_2, \ldots, S_k$
foreach $\theta \in \Theta$
  for $i = 1 \ldots k$
    $h_{i,\theta} = A(S \setminus S_i; \theta)$
  $\text{error}(\theta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\theta})$
output
  $\theta^\star = \text{argmin}_\theta \left[\text{error}(\theta)\right]$
  $h_{\theta^\star} = A(S; \theta^\star)$

What if learning fails?

You use training data S and validation to pick a model h_S...
everything looks good!
But then, on test set results are bad...

What can we do?

Need to understand where the error comes from!

Two cases:

- $L_S(h_s)$ is large
- $L_S(h_s)$ is small

$L_S(h_s)$ is large

Let $h^* \in \arg\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$.

Note that:

$$L_S(h_S) = (L_S(h_S) - L_S(h^*)) + (L_S(h^*) - L_{\mathcal{D}}(h^*)) + L_{\mathcal{D}}(h^*)$$

and

- $L_S(h_S) - L_S(h^*) < 0$
- $L_S(h^*) \approx L_{\mathcal{D}}(h^*)$

Therefore:

$L_S(h_S)$ large $\Rightarrow$ $L_{\mathcal{D}}(h^*)$ is large $\Rightarrow$ approximation error is large

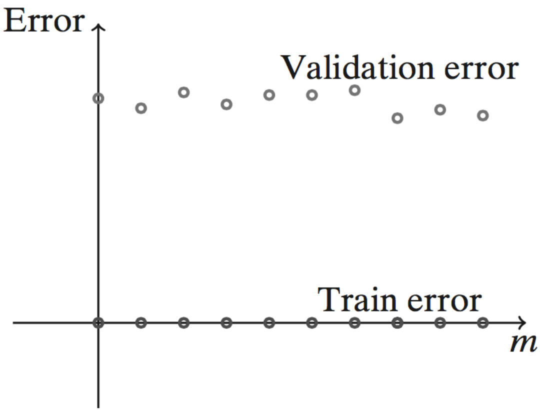
$L_S(h_S)$ is small

Need to understand if $L_{\mathcal{D}}(h^*)$ is large or not!

How?

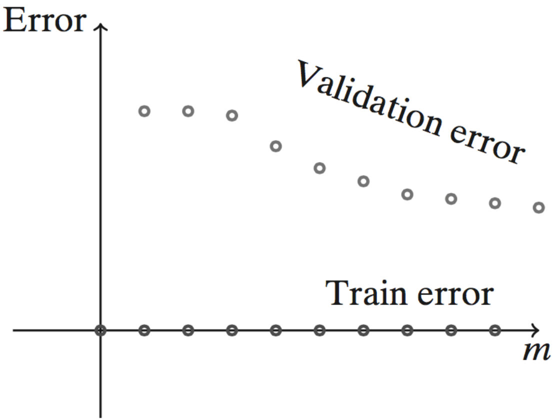
Learning curves: plot of training error and validation error when we run our algorithms on *prefixes of the data of increasing size* m

Case 1



$\Rightarrow$ There is no evidence that the approximation error of $\mathcal{H}$ is good (i.e., that is small)

Case 2



$\Rightarrow \mathcal{H}$ may have a good approximation error but maybe we do not have enough data

Summarizing

Some potential steps to follow if learning fails:

- if you have parameters to tune, plot model-selection curve to make sure they are tuned appropriately
- if training error is excessively large consider:
  - enlarge $\mathcal{H}$
  - change $\mathcal{H}$
  - change feature representation of the data
- if training error is small, use learning curves to understand whether problem is approximation error (or estimation error)
  - if approximation error seems small:
    - get more data
    - reduce complexity of $\mathcal{H}$
  - if approximation error seems large:
    - change $\mathcal{H}$
    - change feature representation of the data

Bibliography

[UML] Chapter 11