Machine Learning

Model Selection and Validation

Fabio Vandin

November 7, 2017
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

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

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

Note: can be done using the linear regression machinery we have seen... how?

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!

We will see 2 approaches:

- model selection using Structural Risk Minimization (SRM)
- validation
SRM can be used to tune tradeoff between bias and complexity

Let $\mathcal{H} = \bigcup_{n \in \mathbb{N}} \mathcal{H}_n$, where each $\mathcal{H}_n$:

- has uniform convergence property
- sample complexity of $\mathcal{H}_n$ is of the type:

$$m_{\mathcal{H}_n}(\epsilon, \delta) \leq \frac{g(n) \log(1/\delta)}{\epsilon^2}$$

where $g : \mathbb{N} \rightarrow \mathbb{R}$ is some monotonically increasing function.

**Example:**

- $\mathcal{H}_d =$ set of polyomials of degree at most $d$, for $d \in \mathbb{N}$
SRM follows a *bound minimization* approach

**Bound:** with probability $\geq 1 - \delta$, for every $d \in \mathbb{N}$ and $h \in \mathcal{H}_d$

\[
L_D(h) \leq L_S(h) + \sqrt{\frac{g(d)(\log(1/\delta) + 2 \log d + \log(\pi^2/6))}{m}}
\]

**SRM rule:** pick $d$ and $h \in \mathcal{H}_d$ minimizing

\[
L_S(h) + \sqrt{\frac{g(d)(\log(1/\delta) + 2 \log d + \log(\pi^2/6))}{m}}
\]

**Note:** upper bound may be pessimistic
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 \( V = (x_1, y_1), \ldots, (x_{m_v}, y_{m_v}) \) be a set of fresh \( m_v \) samples from \( \mathcal{D} \) and let

\[
L_V(h) = \frac{1}{m_v} \sum_{i=1}^{m_v} \ell(h, (x_i, y_i))
\]

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 \( \mathcal{H}_d \)
- \( \text{VCdim}(\mathcal{H}_d) = d \)

Then (by fundamental theorem of learning):

\[
L_D(h) \leq L_S(h) + \sqrt{C \frac{d + \log(1/\delta)}{m}}
\]

where \( C \) is a constant.

From previous proposition:

\[
L_D(h) \leq L_V(h) + \sqrt{\frac{\log(2/\delta)}{2m_v}}
\]

\( \Rightarrow \) 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* set

A similar approach can be used for model selection...
Validation for Model Selection

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

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

Validation set: $\mathcal{V} = (x_1, y_1), \ldots, (x_{m_v}, y_{m_v})$ be a set of fresh $m_v$ samples from $\mathcal{D}$
⇒ choose final hypothesis 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

\[
\forall h \in \{h_1, \ldots, h_r\} : |L_D(h) - L_V(h)| \leq \sqrt{\frac{\log(2r/\delta)}{2m_V}}
\]

**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 ⇒ 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 we choose among many models ($r$ is large)!

**Question:** how can we estimate the true risk after model selection?
Train-Validation-Test Split

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

**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 $L_D(h)$

$\Rightarrow$ the estimate from the test set has the guarantees provided by the proposition on estimate of $L_D(h)$ for 1 class

**Note:**

- the test set is *not involved* in the choice of $h$
- if after using the test set to estimate $L_D(h)$ we decide to choose another hypothesis (*because we have seen $L_D(h)$*)...
  $\Rightarrow$ we cannot use the test set again to estimate $L_D(h)$!
$k$-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 $= \text{average of the estimated errors above}$

**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
\textit{k-Fold Cross Validation for Model Selection}

\textbf{input:}
- training set \( S = (x_1, y_1), \ldots, (x_m, y_m) \)
- set of parameter values \( \Theta \)
- learning algorithm \( A \)
- integer \( k \)

\textbf{partition} \( S \) into \( S_1, S_2, \ldots, S_k \)

\textbf{foreach} \( \theta \in \Theta \)

\textbf{for} \( i = 1 \ldots k \)

\( h_{i,\theta} = A(S \setminus S_i; \theta) \)

\( \text{error}(\theta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\theta}) \)

\textbf{output}

\( \theta^* = \arg\min_{\theta} [\text{error}(\theta)] \)

\( h_{\theta^*} = A(S; \theta^*) \)
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 $h^* \in \arg\min_{h \in \mathcal{H}} L_D(h)$

Note that:

$$L_S(h_S) = (L_S(h_S) - L_S(h^*)) + (L_S(h^*) - L_D(h^*)) + L_D(h^*)$$

and

- $L_S(h_S) - L_S(h^*) < 0$
- $L_S(h^*) \approx L_D(h^*)$

Therefore:

$L_S(h_S)$ large $\Rightarrow$ $L_D(h^*)$ is large $\Rightarrow$ approximation error is large
$L_S(h_S)$ is small

Need to understand if $L_D(h^*)$ 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

There is no evidence that the approximation error of $H$ is good.
⇒ $H$ 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 tune parameters, 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