Lecture 7: Validation and Cross-Validation

Introduction to Learning and Analysis of Big Data
Parameters of learning algorithms

- Many learning algorithms have input parameters.
  -  \(k\)-Nearest-Neighbors: parameter \(k\)
    - Large \(k\): might look too far from new example.
    - Small \(k\): might be too sensitive to noise.
  - Soft-SVM: parameter \(\lambda\) (or \(B\))
    - Guarantee with \(\lambda = \sqrt{\frac{2R_D^2}{B^2m}}\):
      \[
      \mathbb{E}_{S \sim D^m}[\text{err}(h_{\hat{w}_S}, D)] \leq \min_{u: \|u\| \leq B} \ell^h(u, D) + \sqrt{\frac{8R_D^2B^2}{m}}.
      \]
    - Smaller \(B\) (larger \(\lambda\) and margin):
      Might get a high \(\min_{u: \|u\| \leq B} \ell^h(u, D)\). (high approximation error)
    - Larger \(B\) (smaller \(\lambda\) and margin):
      Might get a poor result from the sample. (high estimation error).

- What is the optimal value of the input parameter?
- The optimal value of the parameter depends on the unknown distribution \(D\).
- How should we choose this value in practice?
Validation sample

- **Estimate** the quality of the parameter value using a **validation sample**.
- A **validation sample** $V = \{(x'_1, y'_1), \ldots, (x'_n, y'_n)\}$ is a labeled sample that was not used for training the algorithm.
- $V \sim \mathcal{D}^n$, independently from $S \sim \mathcal{D}^m$.
- Use the error on the validation sample as an estimate of $\operatorname{err}(\hat{h}_S, \mathcal{D})$.

Choosing a parameter using a validation set

**input** A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$,
A validation set $V = \{(x'_1, y'_1), \ldots, (x'_n, y'_n)\}$,
A learning algorithm $A$ with a parameter $\alpha$,
$\Psi = \text{A finite set of possible values for } \alpha$.

**output** $\hat{h}_S : \mathcal{X} \to \mathcal{Y}$,
1: for $\alpha \in \Psi$, do
2: $\hat{h}_\alpha \leftarrow A(S, \alpha)$
3: end for
4: Select $\hat{\alpha} \in \arg\min_{\alpha \in \Psi} \operatorname{err}(\hat{h}_\alpha, V)$.
5: Output $\hat{h}_S = \hat{h}_{\hat{\alpha}}$. 
Validation sample

- We are using $\text{err}(\hat{h}_\alpha, V)$ as an estimate for $\text{err}(\hat{h}_\alpha, D)$.
- How good is this estimate?
- Define $Z_i^\alpha = \mathbb{I}[\hat{h}_\alpha(x'_i) \neq y'_i]$, conditional on $S$ — i.e., treat $S$ as constant
- $Z_i^\alpha$ is a random variable over $\{0, 1\}$.
- What is the probability that $Z_i^\alpha = 1$?

\[ P[Z_i^\alpha = 1|S] = P_{(X, Y) \sim D}[\hat{h}_\alpha(X) \neq Y|S] = \text{err}(\hat{h}_\alpha, D). \]

- $Z_1^\alpha, \ldots, Z_n^\alpha$ are statistically independent.
- Their average is the error on $V$.

\[ \text{err}(\hat{h}_\alpha, V) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[\hat{h}_\alpha(x'_i) \neq y'_i] = \frac{1}{n} \sum_{i=1}^{n} Z_i^\alpha. \]

- How close is the average of $Z_1^\alpha, \ldots, Z_n^\alpha$ to $P[Z_i^\alpha = 1|S]$?
Validation sample

- Define $Z_i^\alpha = \mathbb{I}[\hat{h}_\alpha(x'_i) \neq y'_i]$.
- $\text{err}(\hat{h}_\alpha, D) = P[Z_i^\alpha = 1|S]$, and $\text{err}(\hat{h}_\alpha, V) = \frac{1}{n} \sum_{i=1}^{n} Z_i^\alpha$.
- How close is the average of $Z_1^\alpha, \ldots, Z_n^\alpha$ to $P[Z_i^\alpha = 1|S]$?

Hoeffding’s inequality

Let $Z_1, \ldots, Z_n$ be independent random variables over $\{0, 1\}$, where for all $i \leq n$, $P[Z_i = 1] = p$. Then

$$P\left(\left| \frac{1}{n} \sum_{i=1}^{n} Z_i - p \right| \geq \epsilon \right) \leq 2 \exp(-2\epsilon^2 n).$$

- Conclusion: for any $h \in \mathcal{H}$ that does not depend on $V$,

$$P[|\text{err}(h, V) - \text{err}(h, D)| \geq \epsilon] \leq 2 \exp(-2\epsilon^2 n).$$

- There are $|\Psi|$ possible values of $\alpha$.

$$P[\exists \alpha \in \Psi, |\text{err}(\hat{h}_\alpha, V) - \text{err}(\hat{h}_\alpha, D)| \geq \epsilon] \leq 2|\Psi| \exp(-2\epsilon^2 n).$$
Validation sample

- There are $|\Psi|$ possible values of $\alpha$.

$$P[\exists \alpha \in \Psi, |\text{err}(\hat{h}_\alpha, V) - \text{err}(\hat{h}_\alpha, D)| \geq \epsilon] \leq 2|\Psi| \exp(-2\epsilon^2 n).$$

- Conclusion: For any $\delta \in (0, 1)$, if $n \geq \frac{\log(2|\Psi|/\delta)}{2\epsilon^2}$, then

$$P[\forall \alpha \in \Psi, |\text{err}(\hat{h}_\alpha, V) - \text{err}(\hat{h}_\alpha, D)| \leq \epsilon] \geq 1 - \delta.$$ 

- With probability $1 - \delta$, the error estimations on $V$ are close to the true errors on $D$.

- So choosing $\alpha$ based on the validation set is OK.
The holdout set

- In practice, have a single labeled set.
- We need to use some of it for validation.
- Solution: a **holdout set**.

Learning with parameters using a holdout set

**Input** A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, A learning algorithm $A$ with a parameter $\alpha$, $\Psi = A$ finite set of possible values for $\alpha$.

**Output** $\hat{h}_S : \mathcal{X} \rightarrow \mathcal{Y}$,

1: Split the training sample into $S'$ and $V$
2: **for** $\alpha \in \Psi$, **do**
3: $\hat{h}_\alpha \leftarrow A(S', \alpha)$
4: **end for**
5: Select $\hat{\alpha} \in \arg\min_{\alpha \in \Psi} \text{err}(\hat{h}_\alpha, V)$.
6: Run $\hat{h}_S \leftarrow A(S, \hat{\alpha})$.
7: Output $\hat{h}_S$. 
The holdout set

- We use a holdout set for selecting $\alpha$,
- But then run the learning algorithm again on the entire $S$.
- Why don’t we run on the entire $S$ from the beginning?
  - Can’t estimate the error if we used $V$ to learn $\hat{h}_\alpha$!
- There are no simple guarantees anymore:
  - $\text{err}(\hat{h}_\alpha, V)$ is close to $\text{err}(\hat{h}_\alpha, D)$,
  - But is $\hat{h}_S = \mathcal{A}(S, \hat{\alpha})$ similar to $\hat{h}_{\hat{\alpha}} = \mathcal{A}(S', \hat{\alpha})$?
Cross-validation

- Can we improve the accuracy of selecting $\alpha$?
- Use several holdout sets and average the error estimations.
- Holdout sets are created by splitting the training sample several times.

### Learning with parameters using $k$-fold cross-validation

**Input** A training sample $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$,
A learning algorithm $A$ with a parameter $\alpha$,
$\Psi = $ a finite set of possible values for $\alpha$,
$k = $ number of folds.

**Output** $\hat{h}_S : \mathcal{X} \rightarrow \mathcal{Y}$.

1: Split the training sample into $k$ equal parts, $S_1, \ldots, S_k$.
2: for $\alpha \in \Psi$, do
3: for $i \in \{1, \ldots, k\}$ do
4: $V \leftarrow S_i$, $S' \leftarrow S \setminus S_i$
5: $\hat{h}_i \leftarrow A(S', \alpha)$
6: $\epsilon_i \leftarrow \text{err}(\hat{h}_i, V)$.
7: end for
8: $\epsilon_\alpha = \frac{1}{k} \sum_{i=1}^k \epsilon_i$
9: end for
10: Select $\hat{\alpha} \in \arg\min_{\alpha \in \Psi} \epsilon_\alpha$.
11: Run $\hat{h}_S \leftarrow A(S, \hat{\alpha})$.
12: Output $\hat{h}_S$. 
Summary: Selecting parameters for learning algorithms

- Many learning algorithms have parameters.
- The optimal parameter value depends on the unknown distribution.
- To select a good parameter value,
  1. Use parts of the training sample as validation sets
  2. For each validation set, run the learning algorithm on the rest of the training sample
  3. Use the errors on the validation sets to select a good parameter value
  4. Run the algorithm on the entire training sample with the selected value.

- Holdout method: Use a single validation set
- \(k\)-fold cross validation: Use \(k\) validation sets.
  - Standard values of \(k\): 5 or 10.
  - Extreme case: \(k = m\) (sample size). This is called leave one out.
  - Larger \(k\): result is more accurate, but running time is larger.