CSE 471: MACHINE LEARNING

Learning from Examples (Continued)

Outline

- Model Selection and Optimization
- The Theory of Learning
- Nonparametric Models
- Developing Machine Learning Systems
  - Self study, just read through

Model Selection and Optimization

Stationary assumption

- $P(E_i) = P(E_{i+1}) = P(E_{i+2}) = \ldots.$
- $P(E_i) = P(E_i \mid E_i, E_i, E_i, \ldots.)$
- i.i.d - Independent and Identically distributed

Optimal Fit

- Minimize the error rate on Test set
- Suppose a researcher
  - Generates a hypotheses for one setting of hyperparameter
  - Measures the error rates on the <u>test set</u>, and then tries different hyperparameters.
  - No individual hypothesis has peeked at the test set data, but the overall process did, through the researcher.

Optimal Fit

- We need 3 datasets
  - Training set
    - Train the models
  - Validation set (Development set)
    - Evaluate candidate models
    - Choose the best one
  - Test set
    - Final unbiased evaluation of the chosen model

Optimal Fit

- Alternate approach
  - *k*-fold cross validation
    - $k = 5$
    - $k = 10$
    - $k = n$, *Leave-one-out Cross Validation (LOOCV)*
  - We can do without the validation set
  - We still need the test set
-

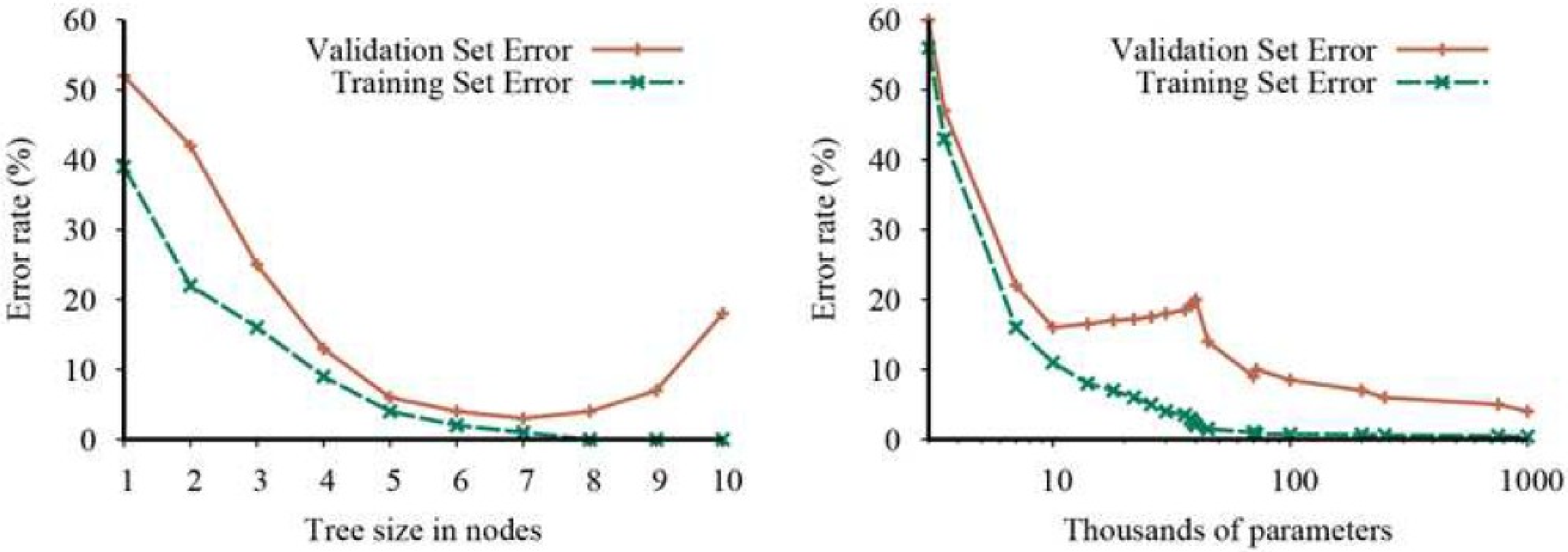
Model Selection

```
function MODEL-SELECTION(Learner, examples, k)
         returns a (hypothesis, error rate) pair
  err ← an array, indexed by size, storing validation-set error rates
  training_set, test_set ← a partition of examples into two sets
  for size = 1 to ∞ do
      err[size] ← CROSS-VALIDATION(Learner, size, training_set, k)
      if err is starting to increase significantly then
          best_size ← the value of size with minimum err[size]
          h ← Learner(best_size, training_set)
          return h, ERROR-RATE(h, test_set)
```

Model Selection

function CROSS-VALIDATION(*Learner*, *size*, *examples*, *k*)
  returns error rate
  $N \leftarrow$ the number of *examples*
  $errs \leftarrow 0$
  for $i = 1$ **to** k **do**
    $validation\_set \leftarrow examples[(i - 1) \times N/k : i \times N/k]$
    $training\_set \leftarrow examples - validation\_set$
    $h \leftarrow Learner(size, training\_set)$
    $errs \leftarrow errs + \text{ERROR-RATE}(h, validation\_set)$
  return $errs / k$ *// average error rate on validation sets,*
                // across k-fold cross-validation

Model Selection



Loss Function

$$L(x, y, \hat{y}) = Utility(\text{result of using } y \text{ given an input } x) - Utility(\text{result of using } \hat{y} \text{ given an input } x)$$

Absolute-value loss: $L_1(y, \hat{y}) = |y - \hat{y}|$

Squared-error loss: $L_2(y, \hat{y}) = (y - \hat{y})^2$

0/1 loss: $L_{0/1}(y, \hat{y}) = 0 \text{ if } y = \hat{y}, \text{ else } 1$

Generalization vs. Empirical Loss

$$GenLoss_L(h) = \sum_{(x,y)\in\varepsilon} L(y, h(x))\, P(x, y)$$

Best Hypothesis

$$h^* = \operatorname*{argmin}_{h\in H} GenLoss_L(h)$$

$$EmpLoss_{L,E}(h) = \sum_{(x,y)\in E} L(y, h(x))\, \frac{1}{N}$$

Estimated Best Hypothesis

$$\hat{h}^* = \operatorname*{argmin}_{h\in H} EmpLoss_{L,E}(h)$$

- ***P(x, y) – Probability of a data point***
- ***ε - Set of all possible data points***

Regularization

$$Cost(h) = EmpLoss(h) + \lambda\, Complexity(h)$$

$$\hat{h}^* = \underset{h \in H}{\operatorname{argmin}}\, Cost(h).$$

- ***Another option is Feature Selection***
  - ***Recursive Feature Elimination (RFE)***
  - ***Correlation study***
  - ***Minimum Redundancy Maximum Relevance (mRMR)***

Hyperparameter tuning

- Hand-tuning
- Grid search
  - Few parameters
  - Each parameter has small number of possible values
  - Can be parallelized
  - if two hyperparameters are independent of each other, they can be optimized separately
- Random search
- Bayesian optimization
- Population-based training (PBT)

Bayesian Optimization

- An ML problem in hyperparameter space!
  - In the validation dataset
- Input
  - The vector of hyperparameter values (**X**)
- Labels
  - A vector of losses (**Y**) on the validation set for the model built with those hyperparameters
  - y is a function of x.
- The learning problem
  - Find the function $f(x)$ that approximates y

Population-based training (PBT)

- First generation of models
  - Use random search of hyperparameters
  - Can be done in parallel
- Second generation of models
  - Hyperparameters from successful (good fit) models from first generation models
  - Mutation
  - Cross-over etc.
  - Can be done in parallel