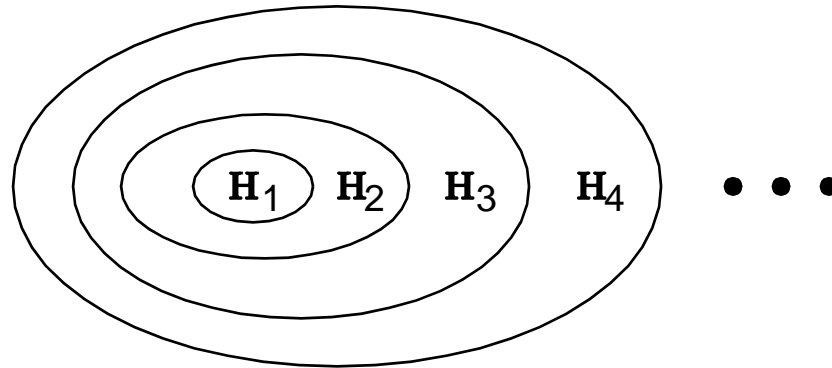


Model Selection and Regularization

Overfitting



$$H_1 < H_2 < H_3 < \dots$$

- If we use an hypothesis space H_i that is too large, eventually we can trivially fit the training data. In other words, the VC dimension will eventually be equal to the size of our training sample m .
- This is sometimes called “model selection”, because we can think of each H_i as an alternative “model” of the data

General Model Selection Problem

Assume that we have a set of models $M = \{M_1, M_2, \dots, M_d\}$ that we are trying to select from. Some examples include:

- **Feature Selection:** each M_i corresponds to using a different set of features from a large set of potential features
- **Algorithm Selection:** e.g., LDA, Naïve, Logistic Regression, DT

 - Each M_i corresponds to a particular algorithm

- **Parameter selection:** e.g., the choice of kernel and C for SVM

 - Each M_i corresponds to a particular choice

Approaches to Model Selection

- Penalty methods
 - Structural Risk Minimization
 - Many others
- Holdout and Cross-validation methods
 - Experimentally determine when overfitting occurs
- Ensembles
 - Full Bayesian methods vote many hypotheses
 $\sum_h P(y|\mathbf{x},h) P(h|S)$
 - Many practical ways of generating ensembles
 - We'll talk about some of these later in the course

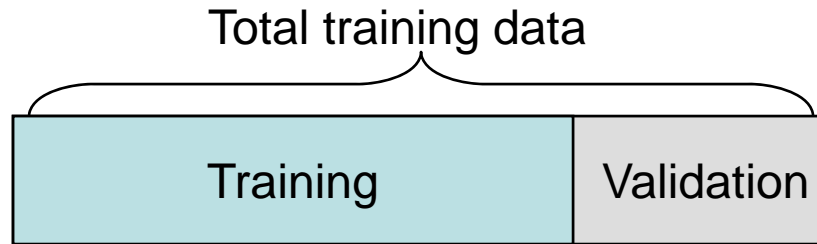
Penalty Methods

- Let $\varepsilon_{\text{train}}$ be our training set error and $\varepsilon_{\text{test}}$ be our test error. Our real goal is to find the h that minimizes $\varepsilon_{\text{test}}$. The problem is that we can't directly evaluate $\varepsilon_{\text{test}}$. We can measure $\varepsilon_{\text{train}}$, but it is optimistic
- Penalty methods attempt to find some penalty such that

$$\varepsilon_{\text{test}} = \varepsilon_{\text{train}} + \text{penalty}$$

- The penalty term is also called a regularizer or regularization term and generally grows with the complexity of the model
- During training, we set our objective function J to be
$$J(\mathbf{w}) = \varepsilon_{\text{train}}(\mathbf{w}) + \text{penalty}(\mathbf{w})$$
and find the \mathbf{w} to minimize this function
- Any examples from what we have seen so far?

Simple Holdout Method



1. Subdivide training set S into S_{train} and $S_{\text{validation}}$
2. Train each model M_i on S_{train} to get some hypothesis h_i
3. Choose and output h_i that gives the best error rate on $S_{\text{validation}}$

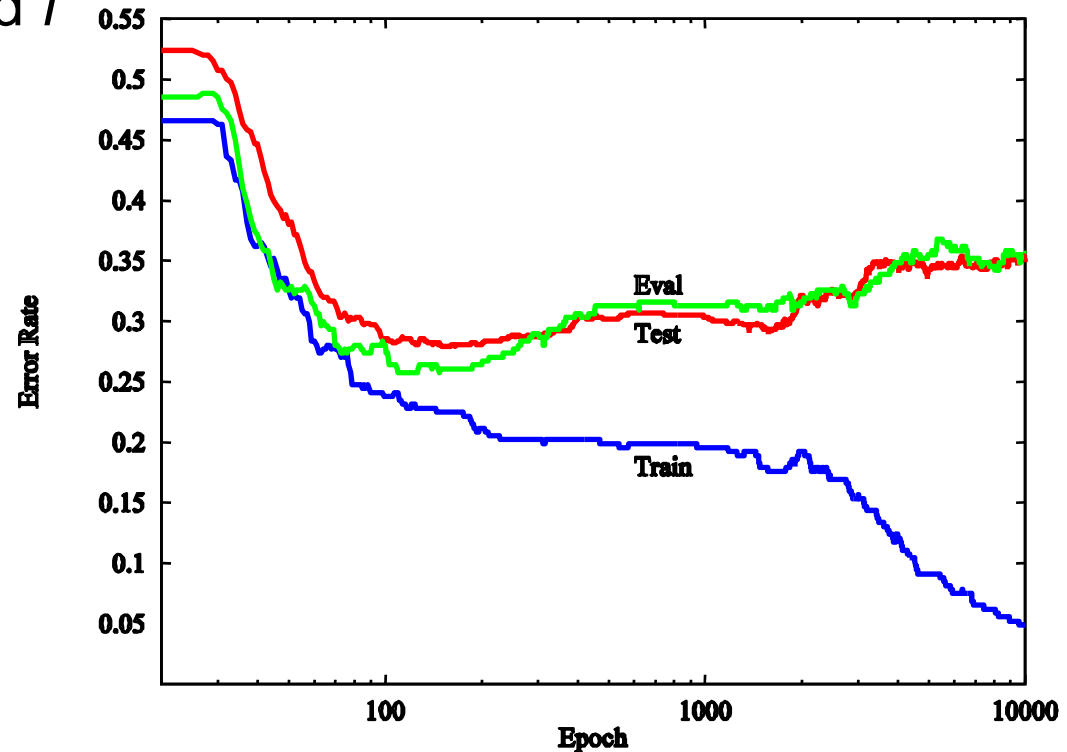
Note:

- Usually 1/3-1/4 of the data is used in the validation set
- Often include an optional step 4
- 4. Retrain M_i on the original training set to get a new h_i
- This often can improve on the original h_i due to larger training set

Example: Select # of Epochs for Neural Net

Example: let M_i represents the choices of training for i epochs

Our goal is to choose a good i



The simple holdout method is widely used to make decisions such as learning rates, number of hidden units, SVM kernel parameters, feature subset selection, etc.

Problem With Simple Holdout Method

- It wastes a good part of the training data
 - The optional step of retraining doesn't solve this problem
 - The model selection choice is still made using only part of the data
 - Still possible to overfit the validation data since it is a relatively small set of data
- To address these problems, we can use a method called **Cross-Validation**

Comments on k-fold Cross-Validation

- A typical value for k is 10
- Computationally more expensive than simple hold-out method but better use of data
- If the data is really scarce, we can use the extreme choice of k equal to the size of the complete training set
 - Each validation set contains only one data point
 - This is referred to as leave-one-out (LOO) cross-validation

Use Cross-Validation For Algorithm Evaluation

- Suppose you developed a new algorithm
- It has a few parameters to tune



Simple hold-out for parameter tuning



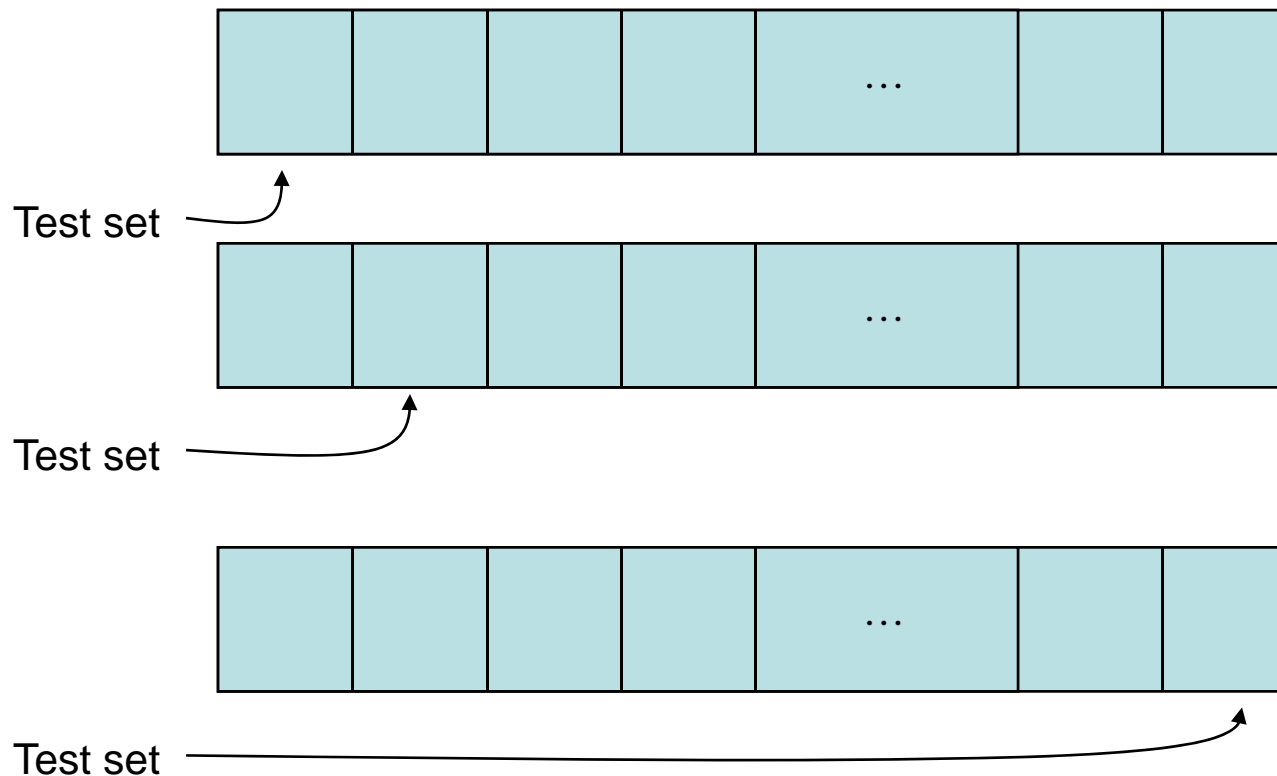
k-fold cross-validation for parameter tuning

All of the data

Never ever ever learn on test data!

Nested Cross-Validation

- Now the reviewer complains that your algorithm is only evaluated on a small test set
- How to address that?



Comments

- Holdout methods are the best way to choose a classifier during learning
 - Reduce error pruning for trees
 - Early stopping for neural networks
- Cross-validation methods are the best way to set a regularization parameter
 - Number k of nearest neighbors in k -NN
 - C and σ for SVMs