Evaluation and Model Selection

Nishant Mehta

Lecture 4 - Part I

Administrivia - Project

Each of you needs to submit a brief, initial proposal by this Friday, September 22nd. Details on how/where to submit are on Brightspace.

The initial proposal consists of:

(1) A suitably informative title.

(2) Two to three sentences describing the project idea.

This is like an elevator pitch: what you would tell someone if you wanted to get them excited about your project but only had 30 seconds to do so.

The initial proposals will be shared with the class on Brightspace, along with your name, email address, and whether you are undergrad/grad.

You all will need to form groups by Friday September 29th. Undergrads team up with undergrads, and grad students team up with grad students. I have set up a forum on Ed Discussion for discussions to help find a group, and you can also email people to find group members or join a group yourself.

Training Error and True Error

Suppose that we have sample *D* of *n* i.i.d. examples drawn according to a distribution *P*.

The sample error of a hypothesis *h* with respect to sample *D* is

$$\hat{R}(h, D) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(x_i))$$

Let $\mathcal{A}$ be a learning algorithm trained on sample *D* and returning hypothesis $\hat{h}_D = \mathcal{A}(D)$. The sample error of $\hat{h}_D$ on *D* is called the ***training error*** of $\mathcal{A}$ and is equal to

also called ***empirical risk***

$$\hat{R}(\hat{h}_D, D) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(y_i, \hat{h}_D(x_i)\right)$$

Training Error and True Error

Let $\mathcal{A}$ be a learning algorithm trained on sample D and returning hypothesis $\hat{h}_D = \mathcal{A}(D)$. The sample error of $\hat{h}_D$ on D is called the ***training error*** of $\mathcal{A}$ and is equal to

also called ***empirical risk***

$$\hat{R}(\hat{h}_D, D) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{h}_D(x_i))$$

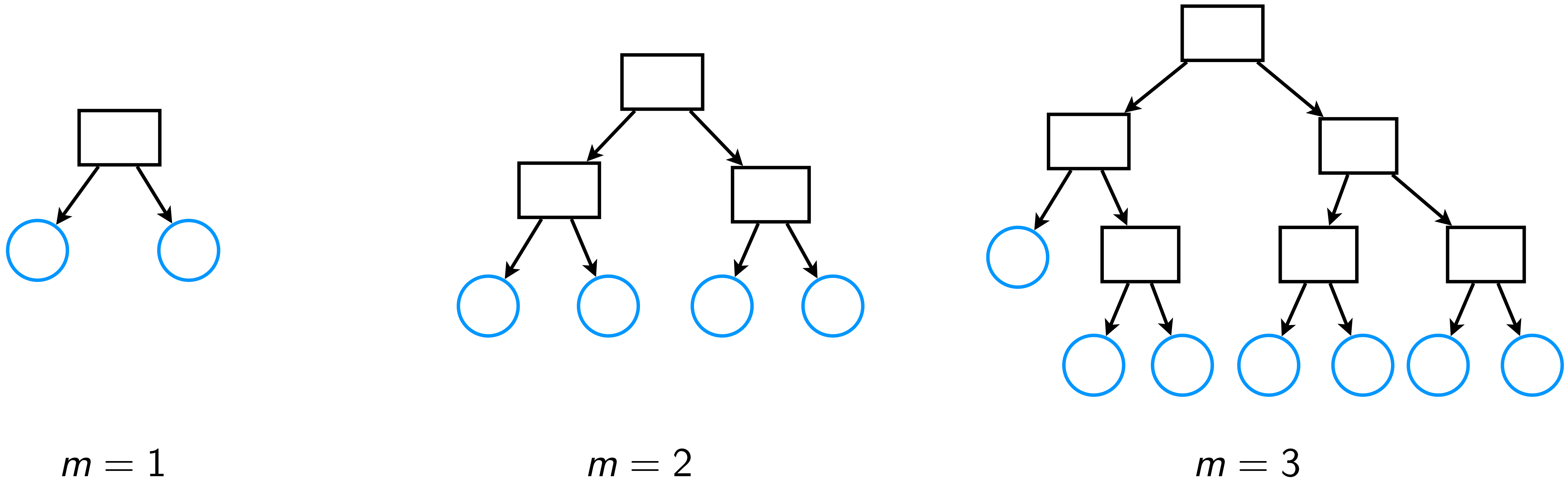
The ***true error*** of the learning algorithm is

$$R(\hat{h}_D) = \mathrm{E}_{(X,Y)\sim P}\left[\ell(Y, \hat{h}_D(X))\right]$$

also called **risk**

Model selection

Suppose that we have M learning algorithms, or ***models***.
For $m = 1, 2, \ldots, M$, algorithm $\mathcal{A}_m$ learns a decision tree with max depth m.



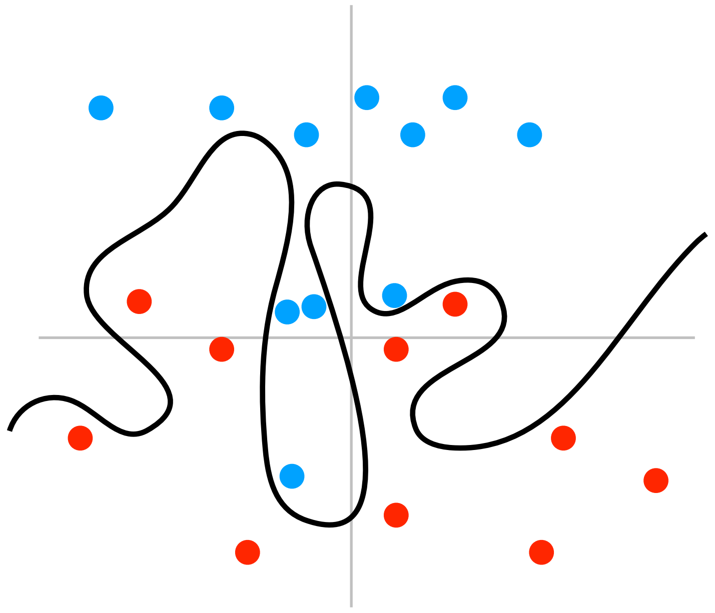
As we increase the learning algorithm index m, we increase the complexity of the model we are using for learning.

How can we select the best model, i.e., the learning algorithm that obtains the lowest true error, when all we can compute is the error on samples?

Model selection

Model selection problem comes up all the time.

Other examples of model selection problems include selecting:



- Degree of polynomial, for regression or separating the input space into two pieces (binary classification)
- Number of trees in a random forest
- How much to prune a decision tree
- Number of nodes in a hidden layer of a neural network (coming up soon)

Hyperparameter tuning

Even after model selection, there is still training to do
(setting the number of trees in a random forest doesn't mean the trees have been learned yet!)

Similarly, there are many other "tuning parameters", or ***hyperparameters***, for which, even after they are set, the main parameters still need to be learned.

Examples of hyperparameters that we will see later in the course:

- Amount of *regularization* in SVM, logistic regression, linear regression
- The *learning rate* for neural network training
- Choice of *k* in nearest neighbor classification

We run into the same question as with model selection:

- How can we select the best hyperparameter configuration,
  i.e., the learning algorithm that obtains the lowest true error,
  when all we can compute is the error on samples?

Estimating true error

How can we estimate the true error of a single hypothesis?

How about training error? $\hat{R}(\hat{h}_D, D)$

No! It's too optimistic. Imagine the algorithm that memorizes the training set and predicts totally randomly on new examples...

Instead, use test error (from an **independent** test sample)

$$\frac{1}{|D_{\text{test}}|} \sum_{(x,y) \in D_{\text{test}}} \ell(y, h(x))$$

This is an ***unbiased*** estimator of the true risk

Estimating true error

How well does the test error estimate the true error?

Suppose we have a classification task, so the losses (errors) are either 0 or 1, and there are ***n*** test examples

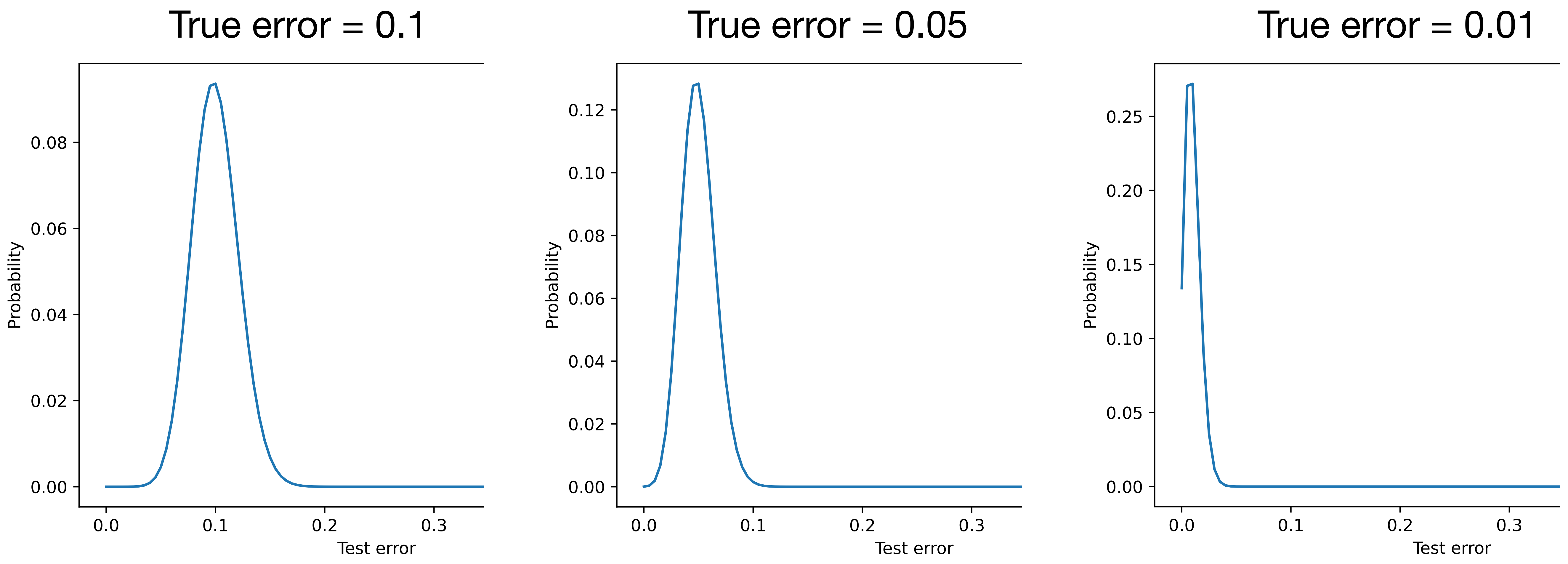
Then, with 95% probability, we have*

$$\left| \hat{R}(\hat{h}_D, D_{\text{test}}) - R(\hat{h}_D) \right| \leq \frac{1.36}{\sqrt{n}}$$

For zero-one loss, test error has a binomial distribution; using this fact can give much better confidence interval

* Where does this come from?
A basic *concentration inequality* called Hoeffding's inequality

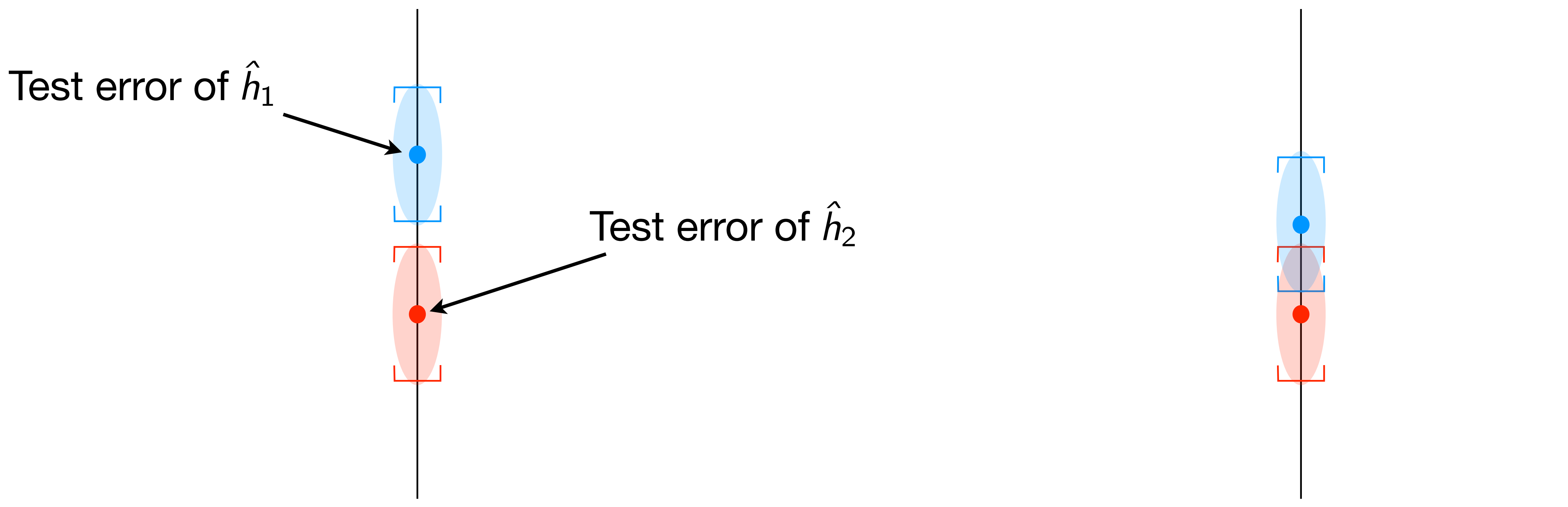
Probability distribution of test error for sample size n = 200



Comparing two hypotheses

How can I compare two hypotheses $\hat{h}_1$ and $\hat{h}_2$ based on their test errors computed from the same test sample?

Approach 1: Form a confidence interval around each hypothesis's true error (using the test error)



Clear separation

Unclear which is better

Comparing two hypotheses

How can I compare two hypotheses $\hat{h}_1$ and $\hat{h}_2$ based on their test errors computed from the same test sample?

Approach 2: Compute difference in test errors

$$\hat{R}(\hat{h}_1, D_{\text{test}}) - \hat{R}(\hat{h}_2, D_{\text{test}}) = \frac{1}{n}\sum_{i=1}^{n}\Big(\ell\big(y_i, \hat{h}_1(x_i)\big) - \ell\big(y_i, \hat{h}_2(x_i)\big)\Big)$$

If 95% confidence interval is entirely positive, then algorithm 2 is better than algorithm 1 (has lower true error)

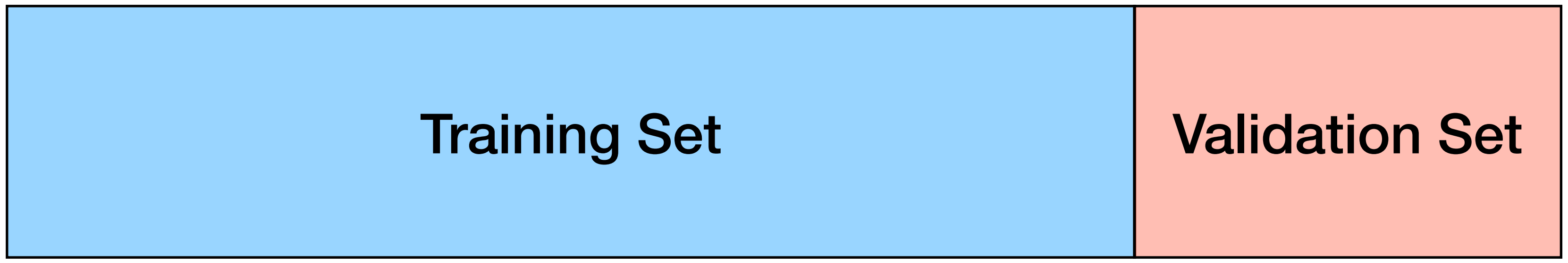
If 95% confidence interval is entirely negative, then algorithm 1 is better than algorithm 2

Comparing learning algorithms

How can I compare the performance of many learning algorithms?

(Basic approach) Cross-validation:

- Split data into training set and ***validation set***



- Run each algorithm $\mathcal{A}_m$ on training set, yielding hypothesis $\hat{h}_m$
- Evaluate validation error of each $\hat{h}_m$ on validation set

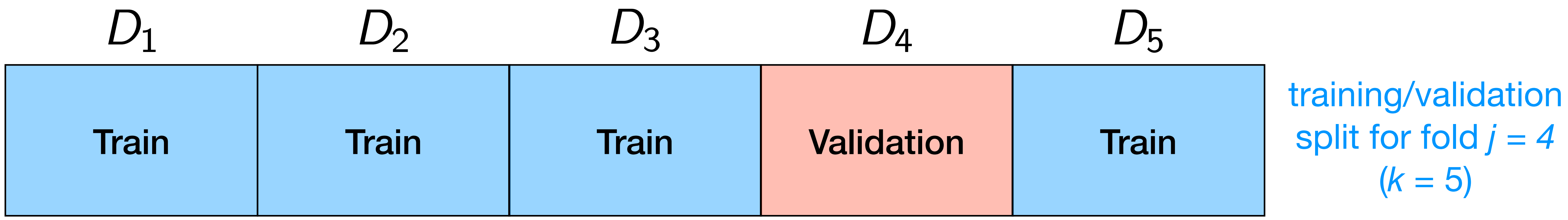
What type of split? 80% training, 20% validation is good rule of thumb

Comparing learning algorithms

How can I compare the performance of many learning algorithms?

(Better approach) *k*-fold cross-validation:

- Split data into *k* pieces (of equal size, ideally)



- In each fold j (for $j = 1$ to k):
  - Run each algorithm $\mathcal{A}_m$ on training set consisting of all parts but the j^{th} part, yielding hypothesis $\hat{h}_{m,j}$.
    Compute test error on j^{th} part, $E_{m,j} = \hat{R}(\hat{h}_{m,j}, D_j)$
- Estimate risk of $\mathcal{A}_m$ using cross-validation error $\frac{1}{k}\sum_{j=1}^{k} E_{m,j}$

Important special cases of k-fold cross-validation

Leave-one-out cross-validation (LOOCV): For dataset of size n, set $k = n$. Each part is one example, hence, during each fold we “leave one out” during training and test on the example that was left out.

$k = 5$: Each part is 20% of the data. Similar to an 80%, 20% training-test split, but with a reduction in variance by averaging over the choice of which 20% we test on.

$k = 10$: Also sensible. As rule of thumb, k between 5 and 10 is generally fine.

Model selection

How can I do model selection (or hyperparameter tuning), and also report the error of the selected model?

First, split the data into training set and test set

Next, do cross-validation on the training set to select the model with the best cross-validation error

For the selected model (algorithm), train the algorithm on the entire training set. Then report the error on the test set.

Very important: We never ever (!) touch the test set until we have selected our model/hyperparameters. So, it is a pristine dataset. Imagine it lives on the moon and we only go to the moon when we are ready to report the test error for our finally selected/tuned learning algorithm.