Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

Topic 3 Model Selection continued

PROF. LINDA SELLIE

Outline

- ❑Motivating example: What polynomial degree should a model use?
- ❑Polynomial transformation

How to create a more complex hypothesis

- ❑Underfitting and overfitting
- ❑Understanding error: Bias and variance and noise
- ❑Learning curves

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

- ❑validation and model selection
- ❑Model selection (with limited data)
- ❑ K-fold cross validation

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

- ❑Regularization

Outline

- ❑Motivating example: What polynomial degree should a e?
- ❑Polynomial transformation
- ❑Underfitting and overfitting
- ❑Understanding error: Bias and varianc
- ❑Learning curves
- ❑validation and model selection
- ❑Model selection (with li
- ❑ K-fold cross validation
- ❑Regularization

Yea!

Uh oh….

How to create a more complex hypothesis

Understanding what wrong

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

Our strategy

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

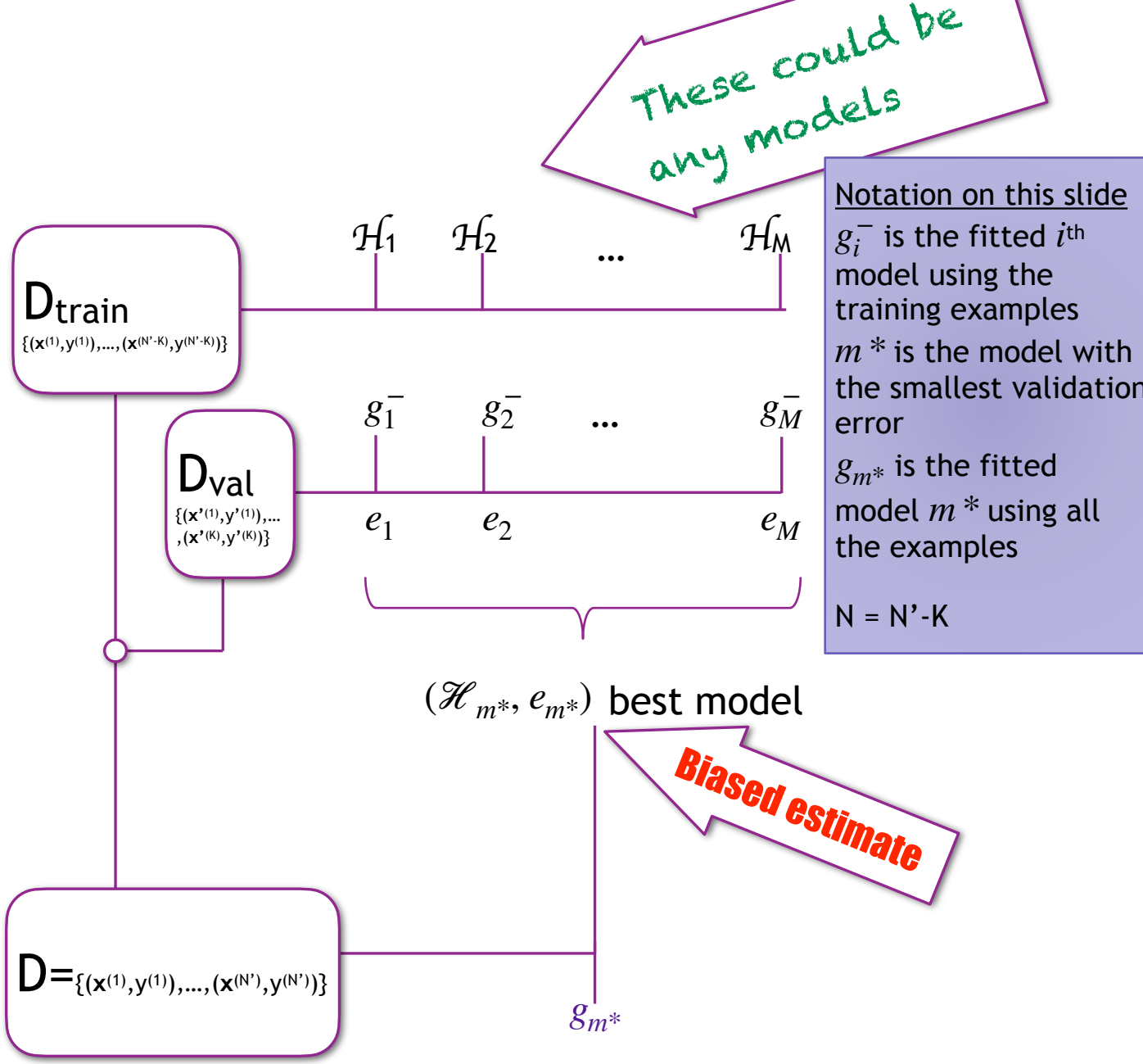
Selecting a model using a validation set

- For each H_i, fit its optimal hypothesis g_i^- using the training set D_{train}
- For each g_i^- estimate the out of sample error e_i using D_{val}
- Pick the H_i that had the smallest e_i
- If we have a test set, we can use that to estimate the error of our hypothesis that had the smallest validation error

If we didn't have a large training set:

- We can then train the selected model using all the data D (i.e. use both D_{train} and D_{val}). Let g_{m*} be the optimal hypothesis found this way.
- If we have a test set, we can use that to estimate the error of g_{m*}

What if we don't have enough data for a test set?



Notation on this slide

g_i^- is the fitted i^{th} model using the training examples

$m*$ is the model with the smallest validation error

g_{m*} is the fitted model $m*$ using all the examples

N = N'-K

Approach taken from pages 19-24 and 143in *Learning from Data*

Thought experiment

1. If we have enough examples in the validation set, is e is a good estimate of E_out?

- yes, it is unbiased
- it is an optimistic estimate, but relatively good estimate
- no, it is not a good estimate

Two hypothesis h_1, h_2

$$E_{\text{out}}(h_1) = E_{\text{out}}(h_2) = \frac{1}{2}$$

Given the error e_1, e_2 estimates for the hypothesis
where we assume (for this thought experiment) that e_1, e_2 is uniform on $[0,1]$

pick $h \in \{h_1, h_2\}$ where $e = \min(e_1, e_2)$

This example comes from Abu-Mostafa

Thought experiment

e_1	e_2	$e = \min\{e_1, e_2\}$
$e_1 > 0.5$	$e_2 > 0.5$	$e > 0.5$
$e_1 < 0.5$	$e_2 > 0.5$	$e < 0.5$
$e_1 > 0.5$	$e_2 < 0.5$	$e < 0.5$
$e_1 < 0.5$	$e_2 < 0.5$	$e < 0.5$

Two hypothesis h_1, h_2

$$E_{\text{out}}(h_1) = E_{\text{out}}(h_2) = \frac{1}{2}$$

Given the error e_1, e_2 estimates for the hypothesis
where we assume (for this thought experiment) that e_1, e_2 is uniform on $[0,1]$

pick $h \in \{h_1, h_2\}$ where $e = \min(e_1, e_2)$

Notice that $E[e] \leq 0.5$

We have an optimistic biased estimate of the error if we estimate

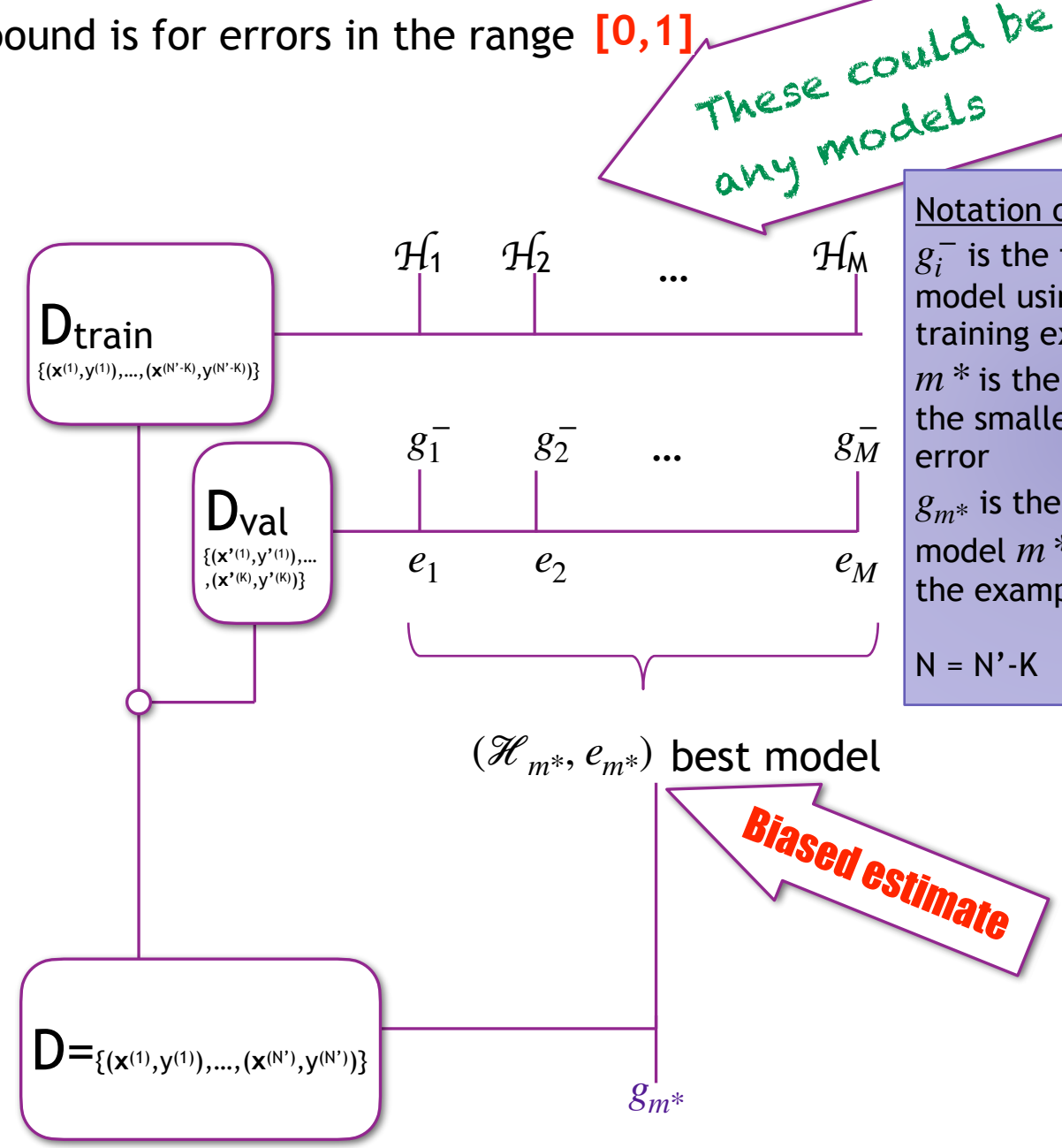
This example comes from Abu-Mostafa

Using validation error to bound out of sample error

The bound is for errors in the range [0,1]

If our validation set has K items, then with probability $1-\delta$

$$E_{\text{out}}(g_{m*}) \leq^{?} E_{\text{out}}(g^{-}_{m*}) \leq \underbrace{E_{\text{val}}(g^{-}_{m*})}_{e_{m*}} + \sqrt{\frac{\ln 2M + \ln\frac{1}{\delta}}{2K}}$$



Notation on this slide

g_i^- is the fitted i^{th} model using the training examples

$m*$ is the model with the smallest validation error

g_{m*} is the fitted model $m*$ using all the examples

N = N'-K

Approach taken from pages 19-24 and 143in *Learning from Data*

http://cs229.stanford.edu/extra-notes/hoeffding.pdf

Outline

- ❑Motivating example: What polynomial degree should a e?
- ❑Polynomial transformation
- ❑Underfitting and overfitting
- ❑Understanding error: Bias and varianc
- ❑Learning curves
- ❑validation and model selection
- ❑Model selection (with limit
- ❑ K-fold cross validation
- ❑Regularization

Yea!

Uh oh….

How to create a more complex hypothesis

Understanding what wrong

Understanding where the error comes from, and how to estimate $E_{\text{out}}[g(\mathbf{x})]$

Our strategy

If we have many different hypothesis classes to choose from - how can we choose wisely? And how can we estimate $E_{\text{out}}[g(\mathbf{x})]$?

Strategies for dealing with a small dataset

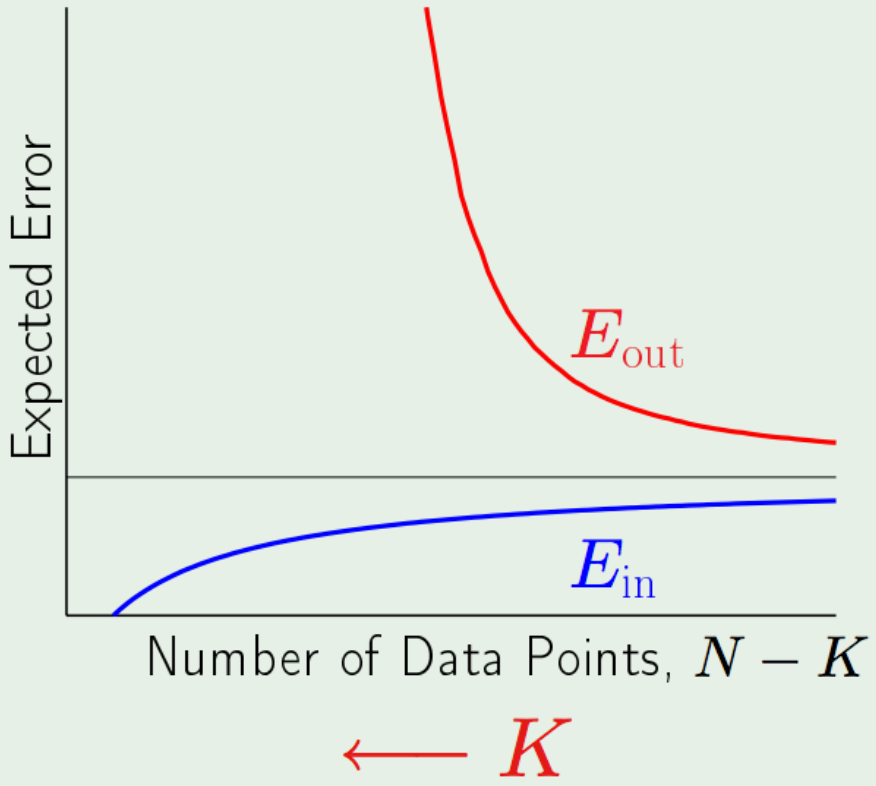
K is taken out of N

Given the data set $\mathcal{D} = (\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \ldots, (\mathbf{x}^{(N)}, y^{(N)})$

$\underbrace{K \text{ points}}_{\mathcal{D}_{\text{val}}} \longrightarrow$ validation $\qquad \underbrace{N - K \text{ points}}_{\mathcal{D}_{\text{train}}} \longrightarrow$ training

Small $K \implies$ bad estimate

Large $K \implies$?

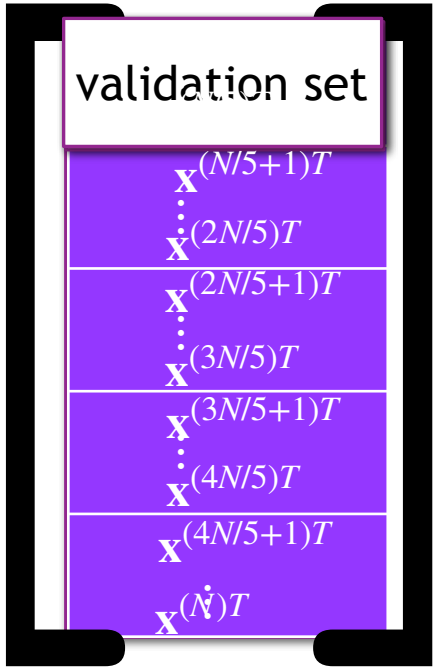


Slide slightly modified from

Choosing the right model using K-fold cross validation

- If there is not enough data, instead of training and validation sets, divide the data into k sets. Commonly K = 5, K = 6, K = 10, or K = N.
- Then train on K-1 of the sets and validate on the remaining set. The estimated error will be the average of the errors.

X =



$e_1 = E_{val}(g_1^-)$

$e_2 = E_{val}(g_2^-)$

$e_3 = E_{val}(g_3^-)$

$e_4 = E_{val}(g_4^-)$

$e_5 = E_{val}(g_5^-)$

not independent error measures

cross validation error

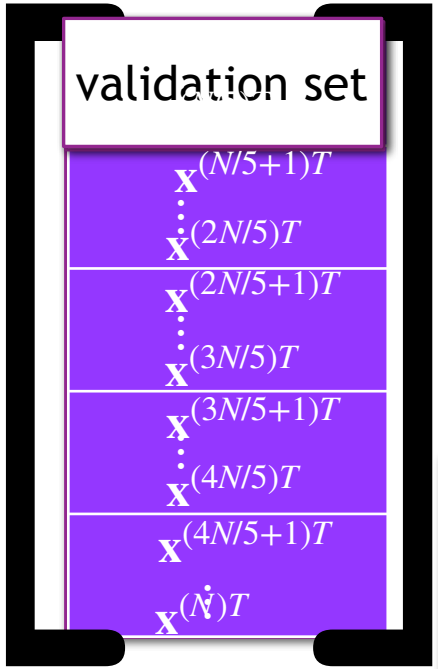
Cross-Validate(D,k)
Divide training data (*randomly*) into K disjoint sets: $D_1, D_2, \ldots, D_K$
For i = 1 to K
    Train on all examples except those in the ith set $D - D_i$
    Let g_i^- be the fitted model
    Validation error $e_i = E_{val}(g_i^-)$
Return $E_{cv} = \frac{1}{K}\sum_{i=1}^{K} e_i$

During this process, you computed K different classifiers. After you are done, run the algorithm again on **all** the data

Choosing the right model using K-fold cross validation

- If there is not enough data, instead of training and validation sets, divide the data into k sets. Commonly K = 5, K = 6, K = 10, or K = N.
- Then train on K-1 of the sets and validate on the remaining set. The estimated error will be the average of the errors.

X =



$e_1 = E_{val}(g_1^-)$

$e_2 = E_{val}(g_2^-)$

$e_3 = E_{val}(g_3^-)$

not independent error measures

E_{cv} is expected error for all data sets of size N-K

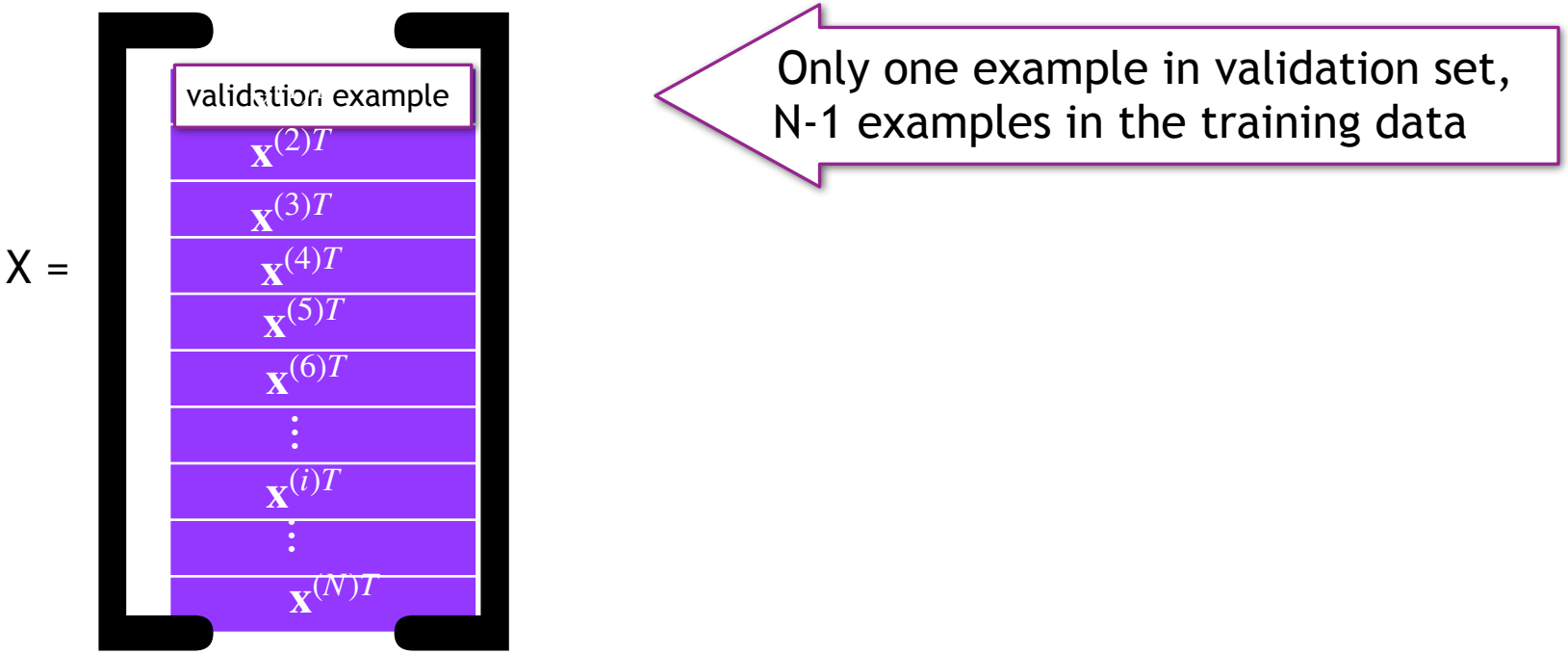
cross validation error

Cross-Validate(D,k)
Divide training data (*randomly*) into K disjoint sets: $D_1, D_2, \ldots, D_K$
For i = 1 to K
    Train on all examples except those in the ith set $D - D_i$
    Let g_i^- be the fitted model
    Validation error $e_i = E_{val}(g_i^-)$
Return $E_{cv} = \frac{1}{K}\sum_{i=1}^{K} e_i$

During this process, you computed K different classifiers. After you are done, run the algorithm again on **all** the data

Leave-one-out cross validation

- If there is very little data, let K = N. This will give the best estimate but the running time may be prohibitive. This is called *leave-one-out cross-validation because* 1 example was left out at a time



During this process, you computed N different classifiers.

Polynomial Features in Scikit-Learn

Generate polynomial transformation of the feature space

"Generate a new feature matrix consisting of all polynomial combinations of the features with degree less than or equal to the specified degree. For example, if an input sample is two dimensional and of the form [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2]."

```
from sklearn.preprocessing import PolynomialFeatures
>>> X = np.arange(6).reshape(3, 2)
>>> X
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> poly = PolynomialFeatures(2)
>>> poly.fit_transform(X)
array([[ 1.,  0.,  1.,  0.,  0.,  1.],
       [ 1.,  2.,  3.,  4.,  6.,  9.],
       [ 1.,  4.,  5., 16., 20., 25.]])
```

Parameters include:
include_bias : *boolean*
The default is True

"Be aware that the number of features in the output array scales polynomially in the number of features of the input array, and exponentially in the degree. High degrees can cause overfitting."

Example from http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

K-Fold Validation in Sklearn

```python
from sklearn import linear_model
import  sklearn.model_selection
```

```python
regr = linear_model.LinearRegression()
```

```python
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)
```

Create a K-fold object

Set shuffle to true

```python
dval = np.arange(1,nfold) #[1,..., nfold-1]
nd = len(dval)
```

Select which degree d models to test

[1 2 3 4 5 6 7 8 9]

```python
# Loop over the folds
MSEval = np.zeros((nd,nfold)) #create a matrix to hold all the values
for isplit, Ind in enumerate(kf.split( X )):
    I_train, I_val = Ind
    X_train = X[I_train]
    y_train = y[I_train]
    X_val = X[I_val]
    y_val = y[I_val]
```

"Generate indices to split data into training and test set."
Ind holds indices for training and validation sets

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  9, 10, 11, 12, 13, 14, 16, 17, 18,
       19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36,
       37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
       54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71,
       72, 73, 74, 77, 78, 79, 80, 81, 83, 84, 85, 86, 88, 89, 92, 93, 94,
       95, 96, 97, 98, 99]), array([ 8, 15, 31, 68, 75, 76, 82, 87, 90, 91]))
```

```python
for it, d in enumerate(dval):
    # polynomial feature transformation
    poly_transformation = PolynomialFeatures(degree=d,include_bias=False)
    X_train_d = poly_transformation.fit_transform(X_train)
    X_val_d = poly_transformation.transform(X_val)
```

If d=1 $\mathbf{x}^{(0)}$=[0.456]
$\Phi_1(\mathbf{x}^{(0)})$= [0.456]
No change (identity transformation)

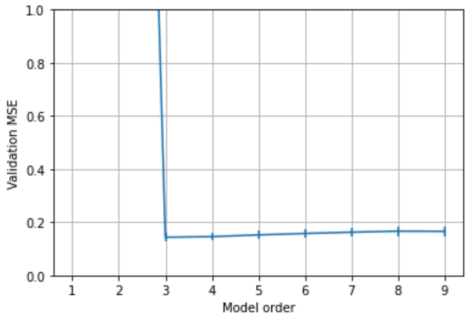
If d = 2,$\mathbf{x}^{(0)}$= [0.456]
$\Phi(\mathbf{x}^{(0)})$=[0.456 0.208]

```python
    # Fit the training data
    regr.fit(X_train_d,y_train)
```

```python
    # Measure MSE on test data
    yhat_val = regr.predict(X_val_d)
    MSEval[it,isplit]= np.mean((yhat_val-y_val)**2)
```

MSEval =

```
[[ 6.77  4.45  7.68  5.44  3.    5.08  4.25  9.32  9.78  5.66]
 [ 6.74  4.42  7.66  5.46  3.02  5.09  4.26 11.03  9.87  5.66]
 [ 0.16  0.23  0.12  0.11  0.06  0.14  0.2   0.2   0.21  0.06]
 [ 0.16  0.23  0.12  0.11  0.05  0.15  0.21  0.19  0.21  0.06]
 [ 0.16  0.23  0.12  0.11  0.05  0.15  0.21  0.19  0.21  0.06]
 [ 0.17  0.23  0.13  0.11  0.05  0.15  0.23  0.2   0.22  0.06]
 [ 0.21  0.24  0.13  0.13  0.06  0.15  0.23  0.22  0.22  0.06]
 [ 0.21  0.25  0.14  0.12  0.06  0.15  0.23  0.22  0.22  0.06]
```




```python
MSE = np.mean(MSEval,axis=1)
```

```
MSE=[5.99764938, 6.17617688, 0.14325349, 0.14562991, 0.15220936, 0.15745814, 0.16233707, 0.16630905, 0.16562408]
```

```python
imin = np.argmin(MSE)
```

The selected model order is 3

K-Fold Validation in Sklearn

```
from sklearn import linear_model
import  sklearn.model_selection
```

```
regr = linear_model.LinearRegression()
```

```
nfold = 10
kf = sklearn.model_selection.KFold(n_splits=nfold,shuffle=True)
```

Create a K-fold object

Set shuffle to true

```
dval = np.arange(1,nfold) #[1,..., nfold-1]
nd = len(dval)
```

Select which degree d models to test

[1 2 3 4 5 6 7 8 9]

```
# Loop over the folds
MSEval = np.zeros((nd,nfold)) #create a matrix to hold all the values
for isplit, Ind in enumerate(kf.split( X )):
    I_train, I_val = Ind
    X_train = X[I_train]
    y_train = y[I_train]
    X_val = X[I_val]
    y_val = y[I_val]
```

"Generate indices to split data into training and test set."
Ind holds indices for training and validation sets

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  9, 10, 11, 12, 13, 14, 16, 17, 18,
       19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 32, 33, 34, 35, 36,
       37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53,
       54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 69, 70, 71,
       72, 73, 74, 77, 78, 79, 80, 81, 83, 84, 85, 86, 88, 89, 92, 93, 94,
       95, 96, 97, 98, 99]), array([ 8, 15, 31, 68, 75, 76, 82, 87, 90, 91]))
```

```
for it, d in enumerate(dval):
    # polynomial feature transformation
    poly_transformation = PolynomialFeatures(degree=d,include_bias=False)
    X_train_d = poly_transformation.fit_transform(X_train)
    X_val_d = poly_transformation.transform(X_val)
```

If d = 3, $\mathbf{x}^{(0)}$= [0.456]
$\Phi_3(\mathbf{x}^{(0)})$=[0.456 0.208 0.095]

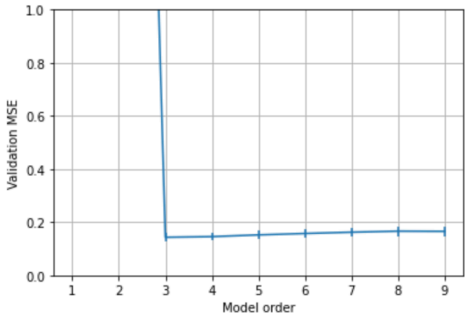
If d = 4, $\mathbf{x}^{(0)}$= [0.456]
$\Phi_4(\mathbf{x}^{(0)})$=[0.456 0.208 0.095 0.043]

```
    # Fit the training data
    regr.fit(X_train_d,y_train)
```

```
    # Measure MSE on test data
    yhat_val = regr.predict(X_val_d)
    MSEval[it,isplit]= np.mean((yhat_val-y_val)**2)
```

MSEval =

```
[[ 6.77  4.45  7.68  5.44  3.    5.08  4.25  9.32  9.78  5.66]
 [ 6.74  4.42  7.66  5.46  3.02  5.09  4.26 11.03  9.87  5.66]
 [ 0.16  0.23  0.12  0.11  0.06  0.14  0.2   0.2   0.21  0.06]
 [ 0.16  0.23  0.12  0.11  0.05  0.15  0.21  0.19  0.21  0.06]
 [ 0.16  0.23  0.12  0.11  0.05  0.15  0.21  0.19  0.21  0.06]
 [ 0.17  0.23  0.13  0.11  0.05  0.15  0.23  0.2   0.22  0.06]
 [ 0.21  0.24  0.13  0.13  0.06  0.15  0.23  0.22  0.22  0.06]
 [ 0.21  0.25  0.14  0.12  0.06  0.15  0.23  0.22  0.22  0.06]
```




```
MSE=[5.99764938, 6.17617688, 0.14325349, 0.14562991, 0.15220936, 0.15745814, 0.16233707, 0.16630905, 0.16562408]
```

```
MSE = np.mean(MSEval,axis=1)
```

```
imin = np.argmin(MSE)
```

The selected model order is 3

Extra Slides

You are not responsible for the following material

Validation estimate

The bound is for errors in the range [a,b]]

$$E_{\text{out}}(g_{m*}) \leq^{?} E_{\text{out}}(g_{m*}^{-}) \leq E_{\text{val}}(g_{m*}^{-}) + O\left(\sqrt{\frac{\ln M}{K}}\right)$$

D_{val} used to choose the best model in

$\{\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M\}$

Can we use E_{m*} to bound the error of g_{m*}^{-}?

Hoeffding inequality (stated without proof) for any sample size K, where each random variable is bounded in [a,b] the probability that a average value, v, of the random variables will deviate from its average μ by more that ϵ is:

$$P\left[|v-\mu| > \epsilon\right] \leq 2e^{-2\epsilon^2 K/(b-a)^2} \text{ for any } \epsilon > 0$$

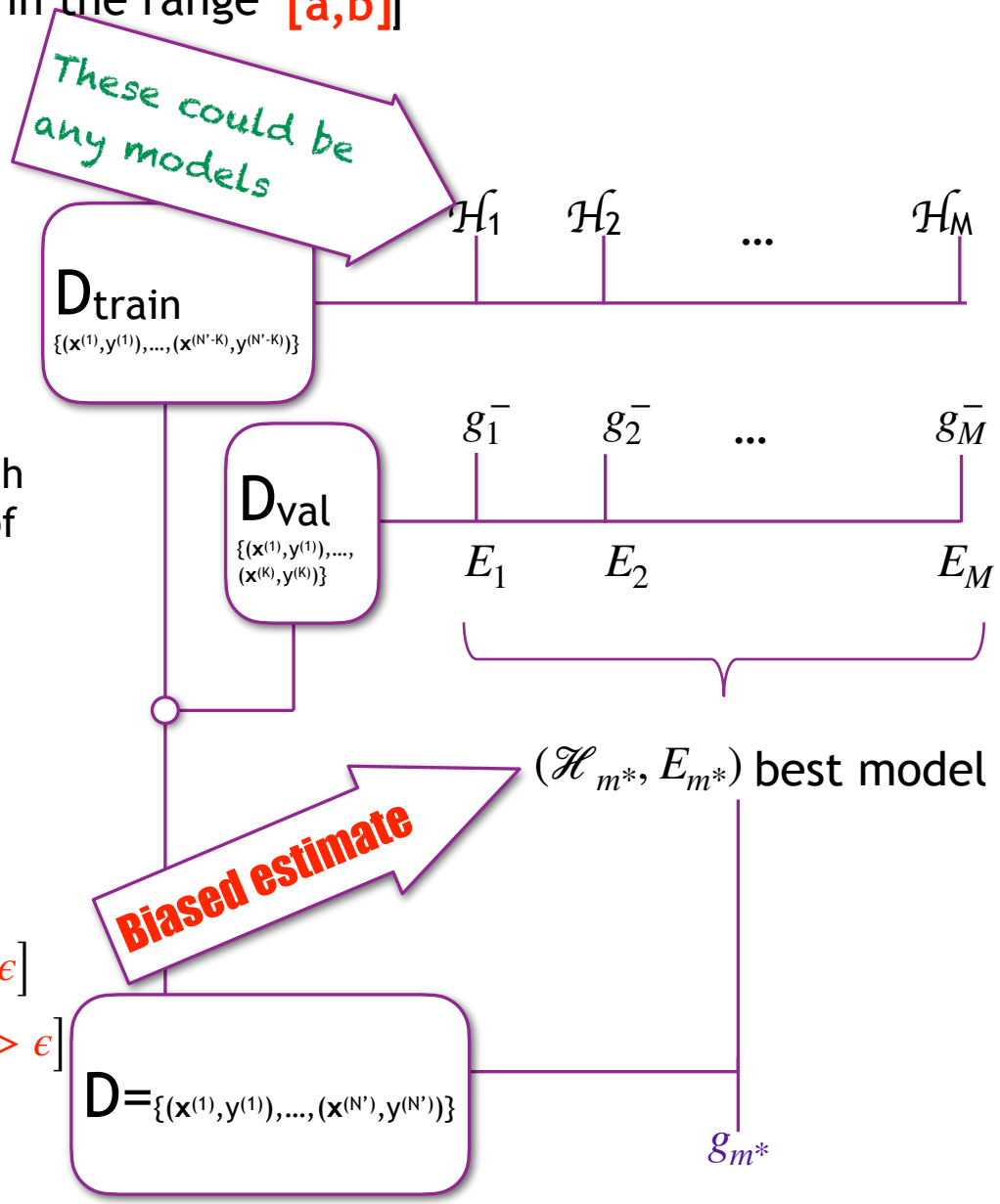
Union bound $P[B_1 \text{ or } B_2 \text{ or } \cdots \text{ or } B_M] \leq P[B_1] + P[B_1] + \cdots + P[B_M]$

Using the *Hoeffding inequality*, *union bound* and M choices (and targets in range [0,1]) we get:

$$P\left[|E_1 - E_{\text{out}}(g_1^{-})| > \epsilon \text{ or } |E_2 - E_{\text{out}}(g_2^{-})| > \epsilon \text{ or } \cdots \text{ or } |E_M - E_{\text{out}}(g_M^{-})| > \epsilon\right]$$

$$< P\left[|E_1 - E_{\text{out}}(g_1^{-})| > \epsilon\right] + P\left[|E_2 - E_{\text{out}}(g_2^{-})| > \epsilon\right] + \cdots + P\left[|E_M - E_{\text{out}}(g_M^{-})| > \epsilon\right]$$

$$\leq M2e^{-2\epsilon^2 K/(b-a)^2} = \delta \text{ for any } \epsilon > 0$$



Validation estimate

The bound is for targets in the range [0, 1]

$$E_{\text{out}}(g_{m*}) \leq^{?} E_{\text{out}}(g^-_{m*}) \leq E_{\text{val}}(g^-_{m*}) + O\left(\sqrt{\frac{\ln M}{K}}\right)$$

$$P\left[|E_1 - E_{\text{out}}(g_1^-)| > \epsilon \text{ or } |E_2 - E_{\text{out}}(g_2^-)| > \epsilon \text{ or } \cdots \text{ or } |E_M - E_{\text{out}}(g_M^-)| > \epsilon\right]$$

$$< P\left[|E_1 - E_{\text{out}}(g_1^-)| > \epsilon\right] + P\left[|E_2 - E_{\text{out}}(g_2^-)| > \epsilon\right] + \cdots + P\left[|E_M - E_{\text{out}}(g_M^-)| > \epsilon\right]$$

$$\leq M 2e^{-2\epsilon^2 K} = \delta \quad \text{for any } \epsilon > 0$$

How can this help us determine a good estimate?

$$M 2e^{-2\epsilon^2 K} = \delta$$

$$e^{-2\epsilon^2 K} = \delta/(2M)$$

$$-2\epsilon^2 K = \log(\delta/(2M))$$

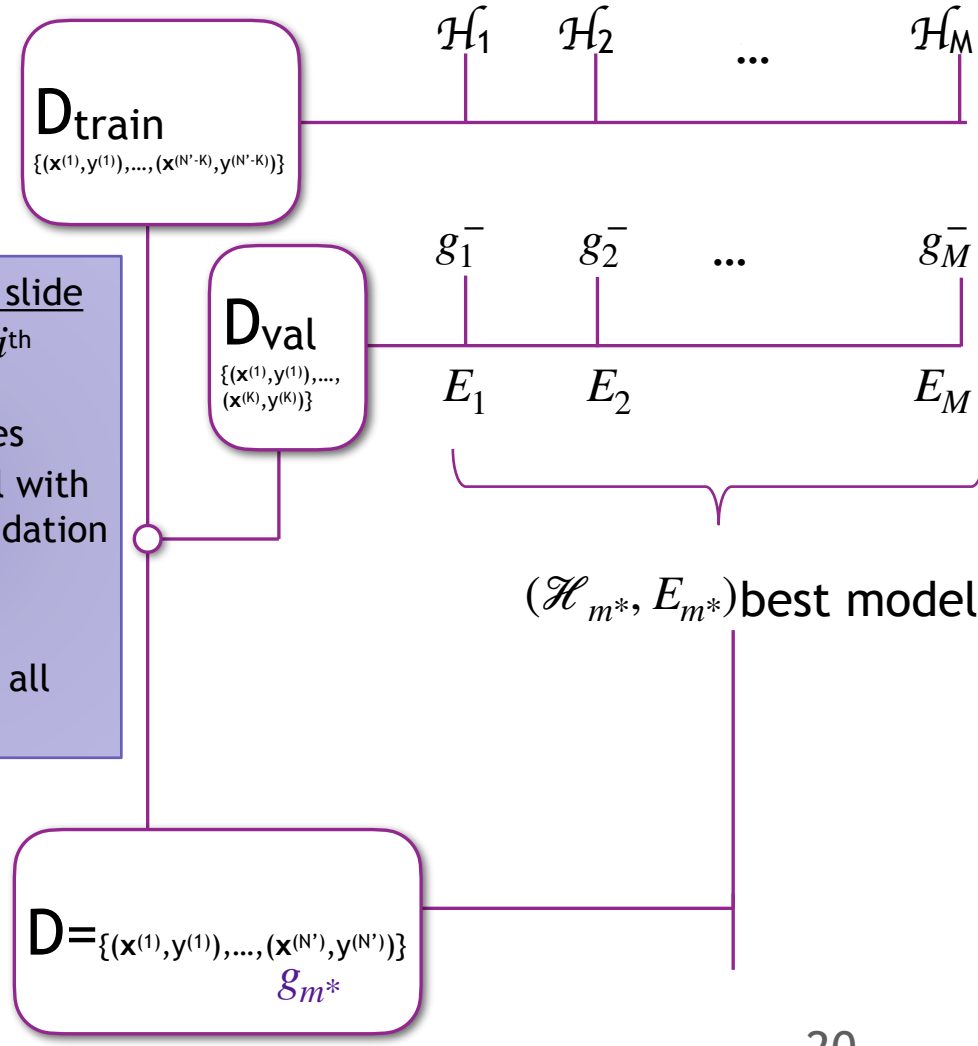
$$\epsilon^2 = -\frac{1}{2K}\log(\delta/(2M))$$

$$\epsilon = \sqrt{-\frac{1}{2K}\log(\delta/(2M))} = \sqrt{\frac{1}{2K}\log(2M/\delta)}$$

With probability $1 - \delta$ we know that

$$E_{\text{out}}(g^-_{m*}) \leq E_{\text{val}}(g^-_{m*}) + O\left(\sqrt{\frac{\ln 2M + \ln\frac{1}{\delta}}{2K}}\right)$$

is $E_{\text{val}}(g^-_{m*}) + O\left(\sqrt{\frac{\ln M}{K}}\right)$



Notation on this slide

g_i^- is the fitted i^{th} model using the training examples

$m*$ is the model with the smallest validation error

g_{m*} is the fitted model $m*$ using all the examples

Example:

Suppose we fit 3 different models g_1^- g_2^- g_3^- using our training data $\{(\mathbf{x}^{(1)},y^{(1)}),\ldots,(\mathbf{x}^{(N-K)},y^{(N-K)})\}$, $y^{(i)} \in \{0,1\}$

For each of our fitted model, we estimated our out of sample error, $E_{\text{out}}(g_i^-)$, using a validation set $E_{val}(g_i^-)$

For each fitted model, we can calculate the probability it's estimate is wrong by more than ϵ using the the *Hoeffding inequality:* $p\left[|E_{out}(g_i^-) - E_{val}(g_i^-)| > \epsilon\right] \leq 2e^{-2\epsilon^2 K}$ for $i = 1,\ldots,3$

What is the probability of <u>**any**</u> of our 3 estimates is wrong by more than ϵ?

$$\text{p}\Big(|E_{\text{out}}(g_1^-) - E_{\text{val}}(g_1^-)| > \epsilon \textbf{ or } |E_{\text{out}}(g_2^-) - E_{\text{val}}(g_2^-)| > \epsilon \textbf{ or } |E_{\text{out}}(g_3^-) - E_{\text{val}}(g_3^-)| > \epsilon\Big)$$

We can get an upper bound of this probability using the *union bound*:

$$\text{p}\Big(|E_{\text{out}}(g_1^-) - E_{\text{val}}(g_1^-)| > \epsilon \textbf{ or } |E_{\text{out}}(g_2^-) - E_{\text{val}}(g_2^-)| > \epsilon \textbf{ or } |E_{\text{out}}(g_3^-) - E_{\text{val}}(g_3^-)| > \epsilon \Big)$$

$$\leq \text{p}\Big(|E_{\text{out}}(g_1^-) - E_{\text{val}}(g_1^-)| > \epsilon\Big) + \text{p}\Big(|E_{\text{out}}(g_2^-) - E_{\text{val}}(g_2^-)| > \epsilon\Big) + \text{p}\Big(|E_{\text{out}}(g_3^-) - E_{\text{val}}(g_3^-)| > \epsilon\Big)$$

$$\leq 3 \cdot 2e^{-2\epsilon^2 K}$$

If we set $\delta = 3 \cdot 2e^{-2\epsilon^2 K}$ then with probability $1 - \delta$ all 3 of our estimates $E_{val}(g_i^-)$ are within ϵ of their out of sample error $E_{\text{out}}(g_i^-)$.

We can rewrite $\delta = 3 \cdot 2e^{-2\epsilon^2 K}$ as $\epsilon = \sqrt{\frac{1}{2K} \log \frac{3 \cdot 2}{\delta}}$

*Given N labeled examples, we randomly split them into two groups D_{train} and D_{val}