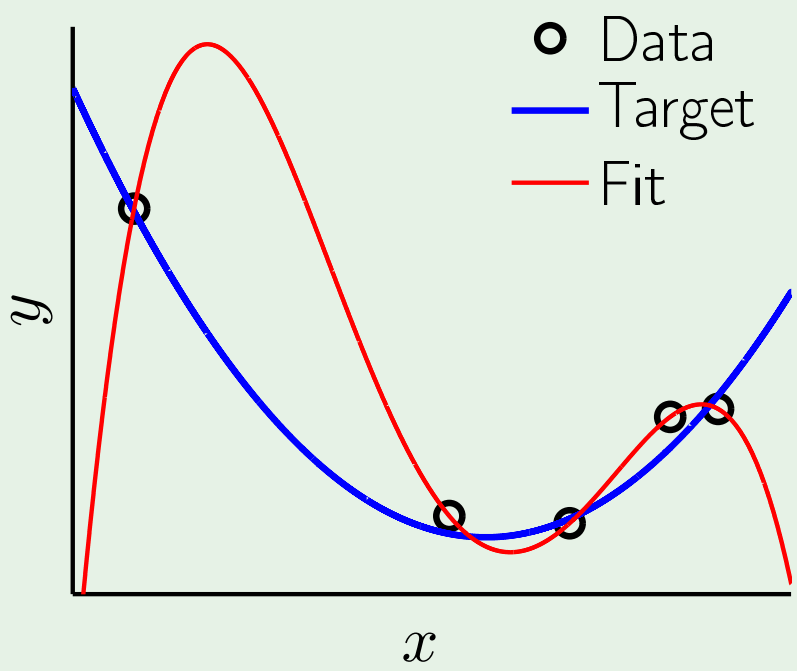
Regularization and model selection

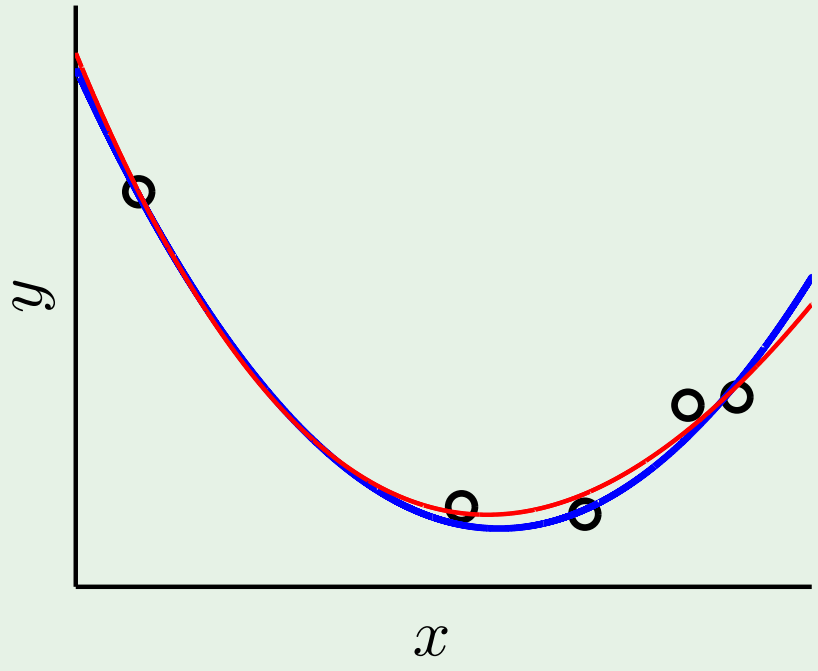
Chapter 4

Regularization

The cure for overfitting - regularization



Without regularization



With regularization

Regularization

How does it work?

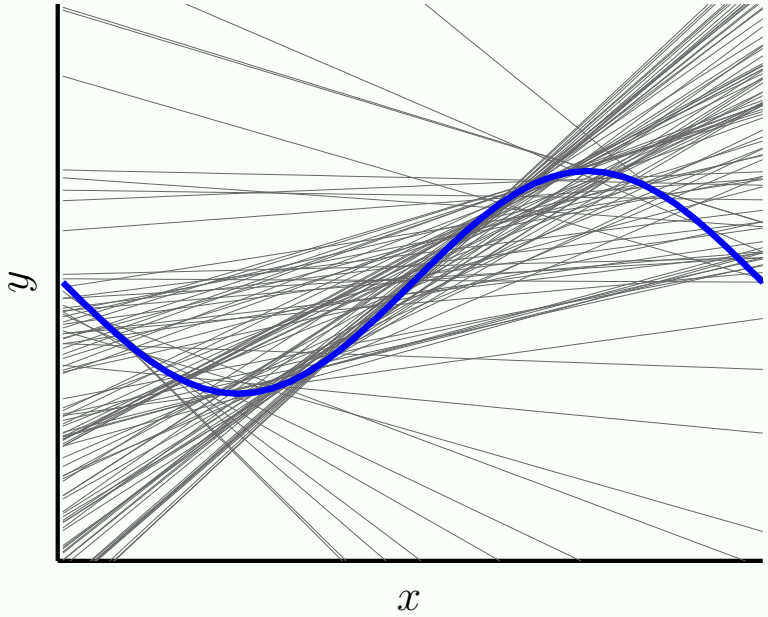
Constrains the model so it cannot fit the noise

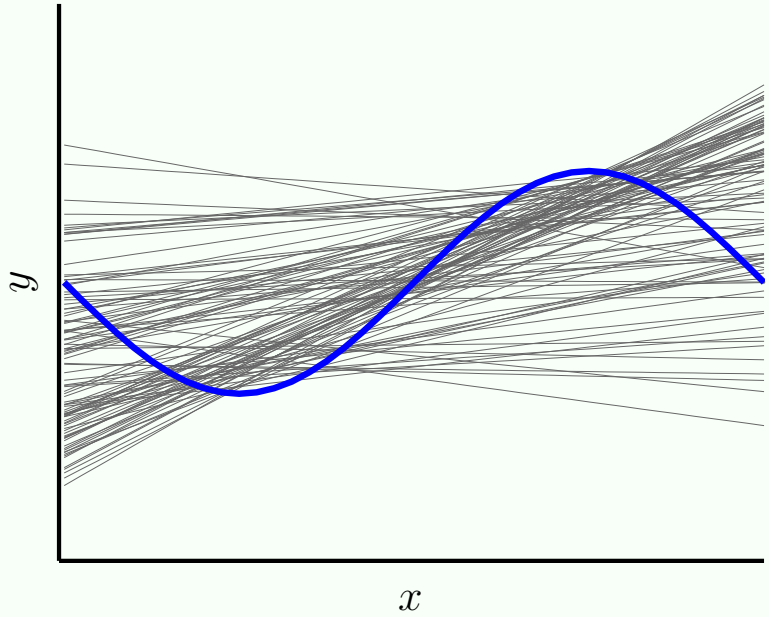
Potential side effect: if it cannot fit the noise, can it fit the target function?

Introduces bias and reduces variance, so that (hopefully) out-of-sample error is lower

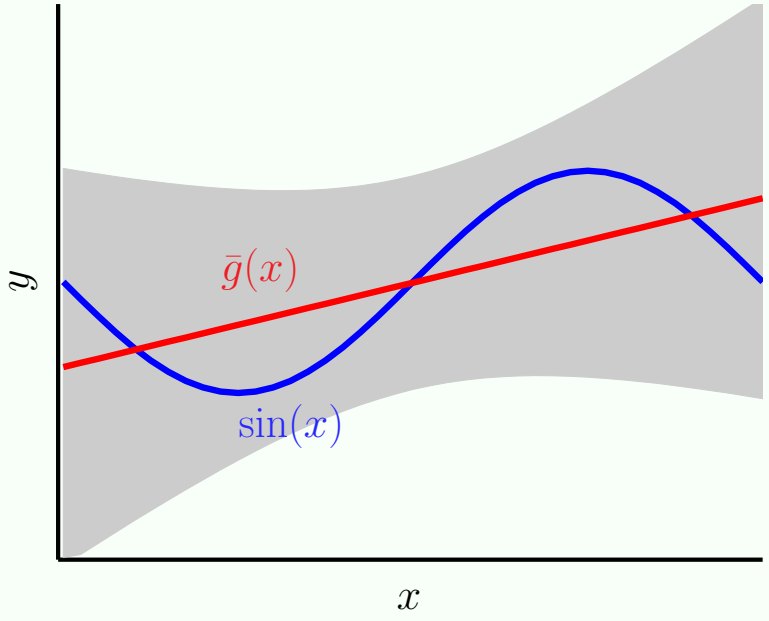
Constraining the model

Let's penalize large weights



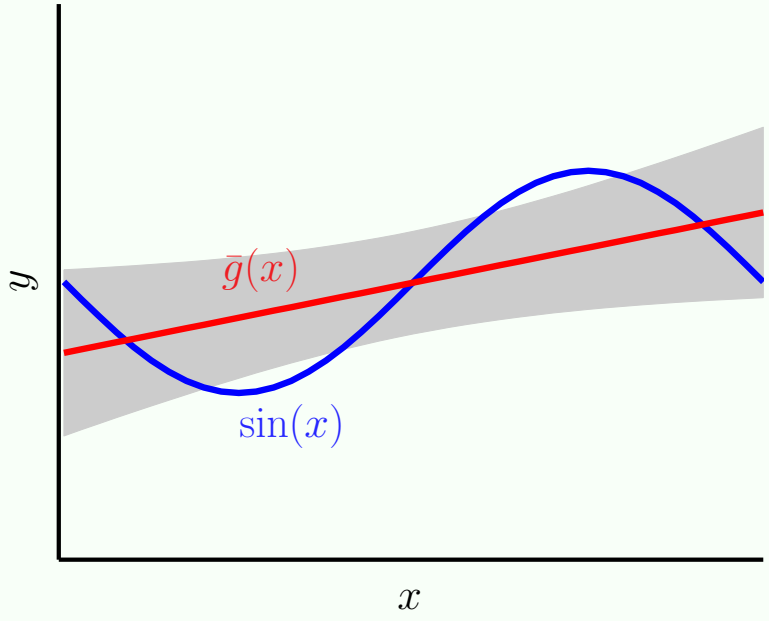


One effect: increased bias



no regularization

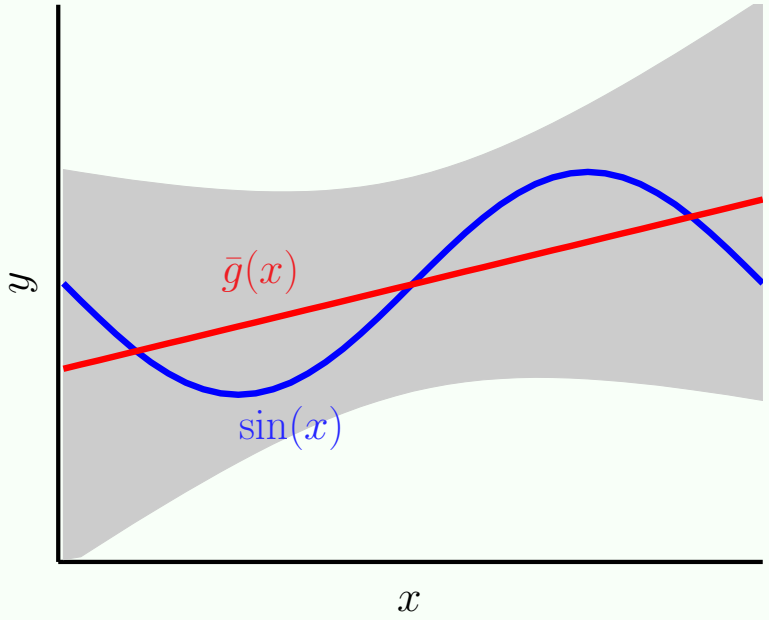
bias $= 0.21$

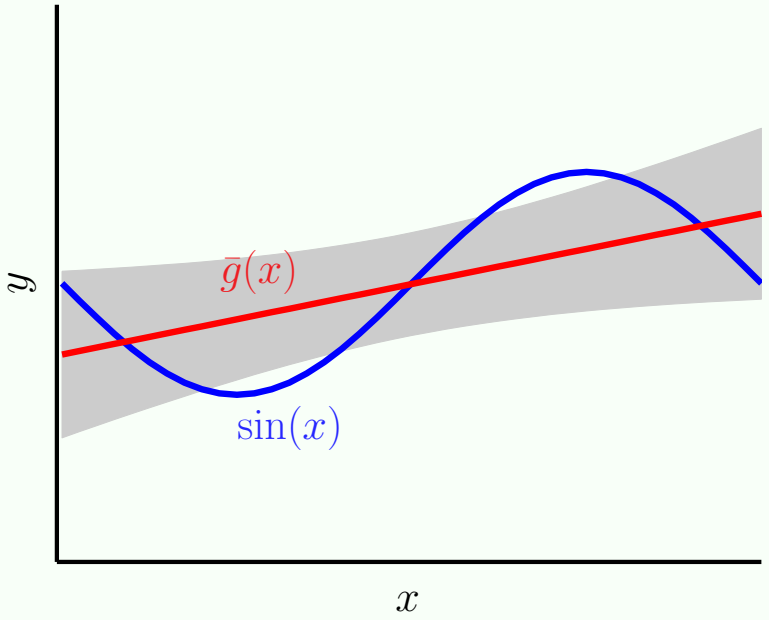


regularization

bias $= 0.23$

Second effect: dramatic reduction in variance





no regularization

bias $= 0.21$

var $= 1.69$

regularization

bias $= 0.23$

var $= 0.33$

Constraining the complexity of the model

Replace E_{in} with:

$$E_{\text{aug}}(h) = E_{\text{in}}(h) + \frac{\lambda}{N}\Omega(h)$$

Regularization term

λ regularization constant

E_{aug} is a better proxy for E_{out} than E_{in}

Choosing a regularizer

We want to constraint the learned function in the direction of the target function.

Intuition: noise is non-smooth

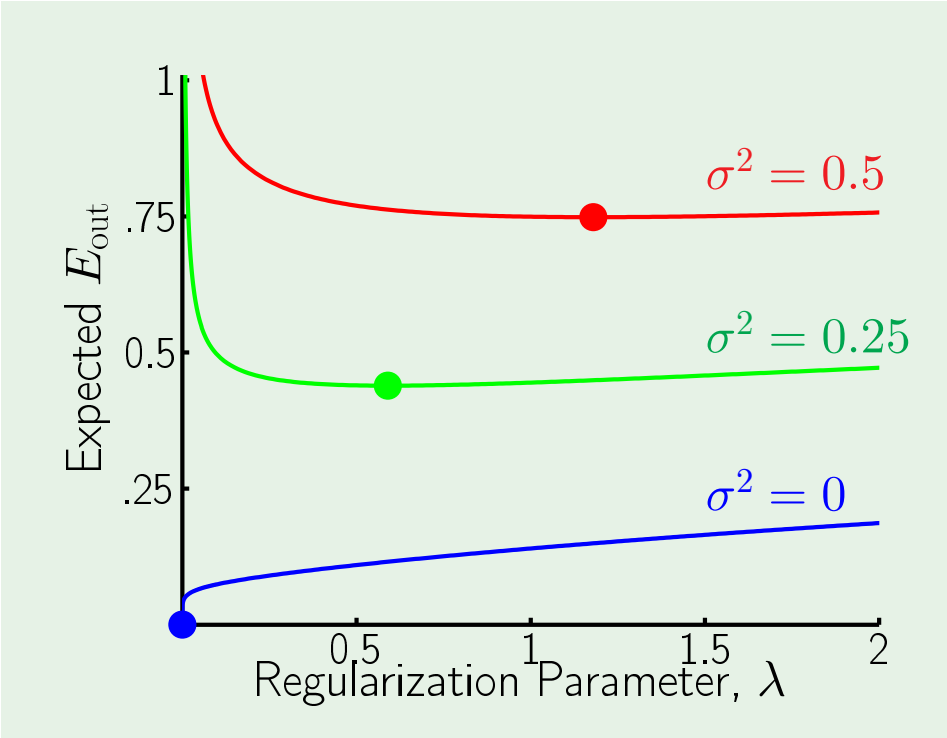
Common choice for the augmented in-sample-error:

$$E_{aug}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda \mathbf{w}^\mathsf{T}\mathbf{w}$$

weight decay regularizer

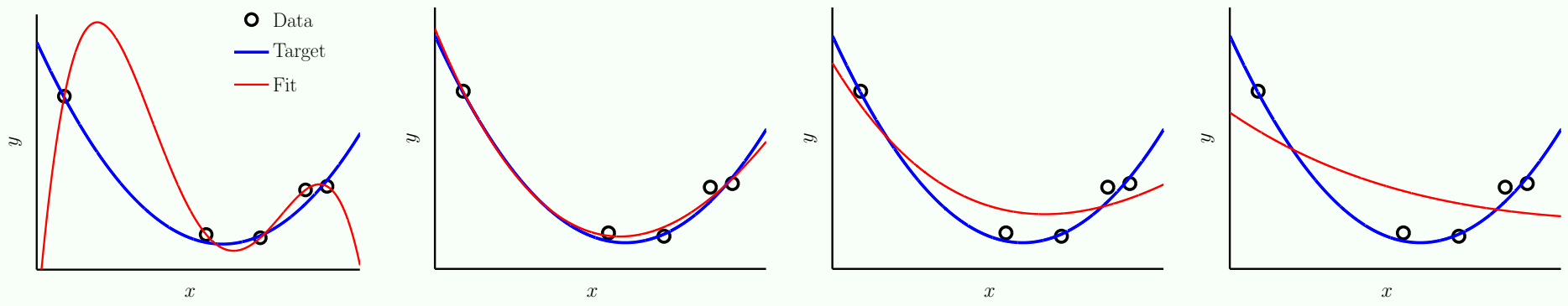
Is there an optimal value for λ?

The behavior of E_{out} as a function of the regularization parameter for varying levels of noise:



Is there an optimal value for λ?

Minimizing $E_{aug}(\mathbf{w}) = E_{in}(\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w}$



Regularized least-squares

Ridge regression:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} (\mathbf{y} - \mathbf{X}\mathbf{w})^\intercal (\mathbf{y} - \mathbf{X}\mathbf{w}) + \frac{\lambda}{2} ||\mathbf{w}||^2$$
$$= (\mathbf{X}^\intercal \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\intercal \mathbf{y}$$

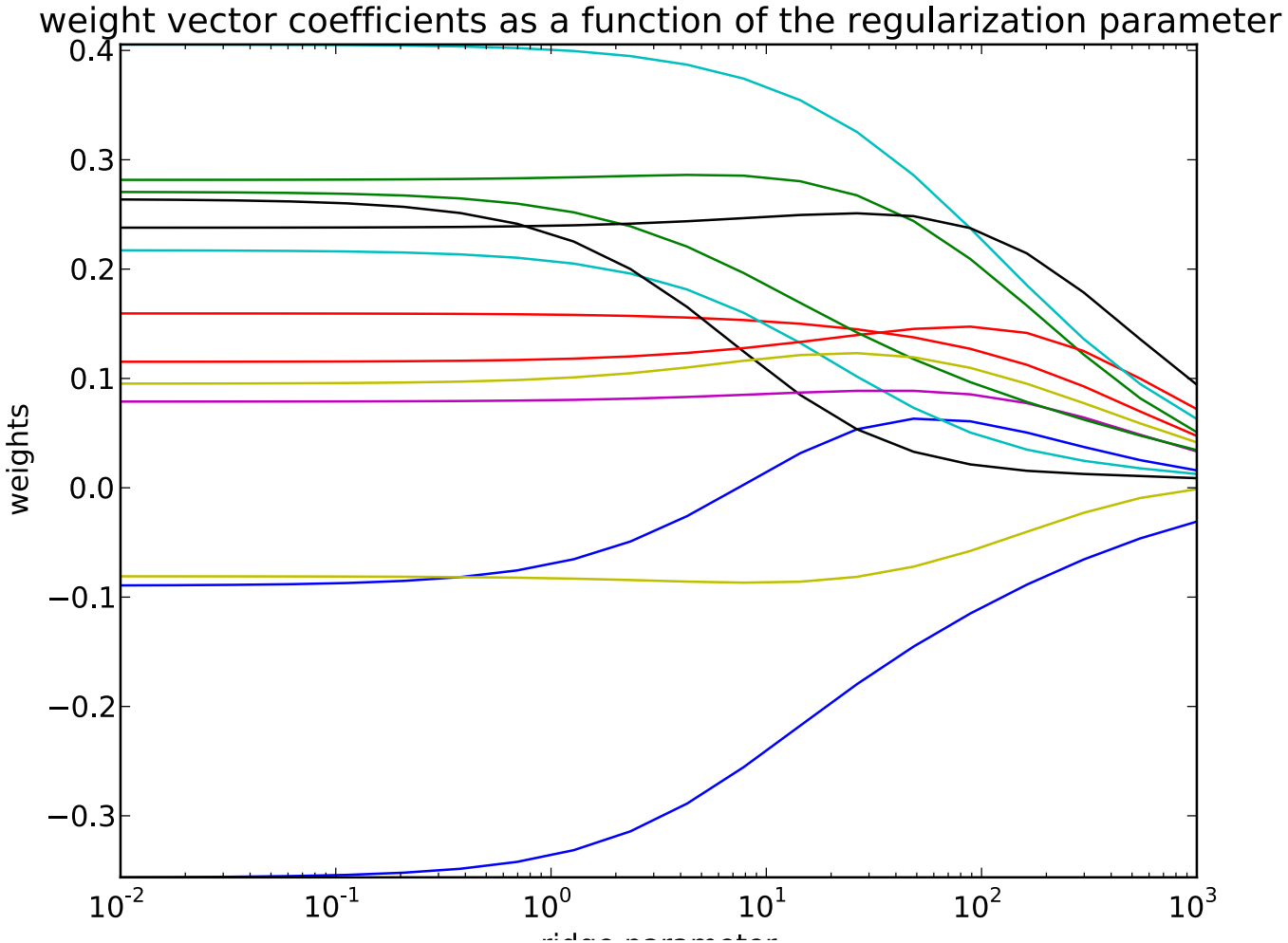
The regularization term controls the size of the components of the weight vector.

There is a tradeoff between fitting (the error term) and regularization. The regularization terms can therefore prevent overfitting. The parameter λ controls this tradeoff.

Many ML methods can be expressed as solution to a criterion of the form:

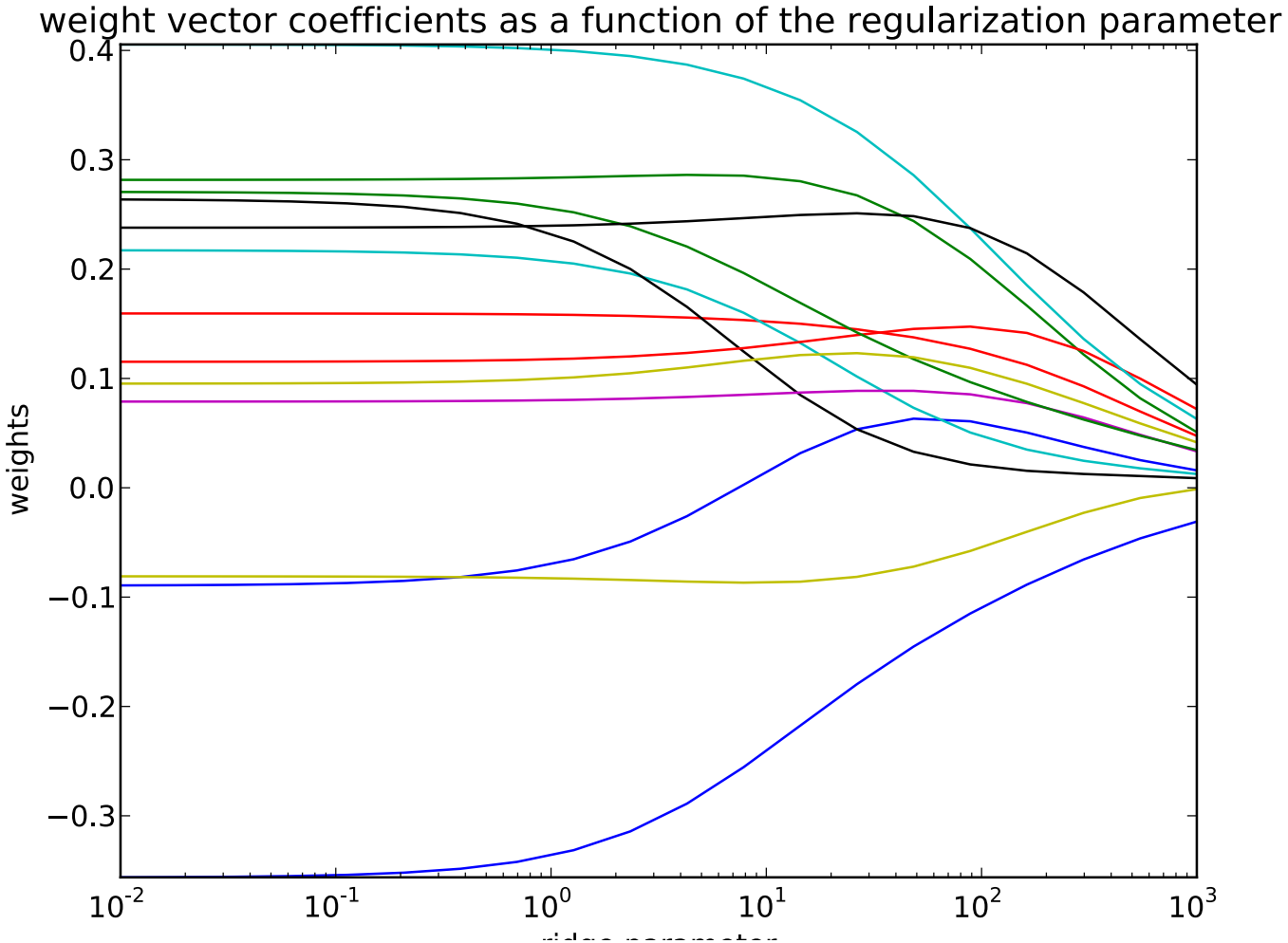
error term + regularization term

The effect of the regularization parameter



Each curve is the magnitude of the weight vector associated with a given feature.
Computed on the scaled version of the "heart" dataset.

The effect of the regularization parameter



As the regularization parameter increases, w_i shrinks toward 0

Assignment 2

Explore the effect of regularization with least-squares regression.

The validation set

How do we choose the value of the regularization parameter?

We take a sneak peak at E_{out} using a validation set.

On a validation set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_K, y_K)$, the error is $E_{\text{val}}(h) = \frac{1}{K} \sum_{k=1}^{K} \mathbf{e}(h(\mathbf{x}_k), y_k)$

Properties of E_{val}

$$\mathbb{E}\left[E_{\text{val}}(h)\right] = \frac{1}{K} \sum_{k=1}^{K} \mathbb{E}\left[\mathbf{e}(h(\mathbf{x}_k), y_k)\right] = E_{\text{out}}(h)$$

$$\text{var}\left[E_{\text{val}}(h)\right] = \frac{1}{K^2} \sum_{k=1}^{K} \text{var}\left[\mathbf{e}(h(\mathbf{x}_k), y_k)\right] = \frac{\sigma^2}{K}$$

$$E_{\text{val}}(h) = E_{\text{out}}(h) \pm O\left(\frac{1}{\sqrt{K}}\right)$$

Section 4.3.1

Choosing the size of the validation set

Given the data set $\mathcal{D} = (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

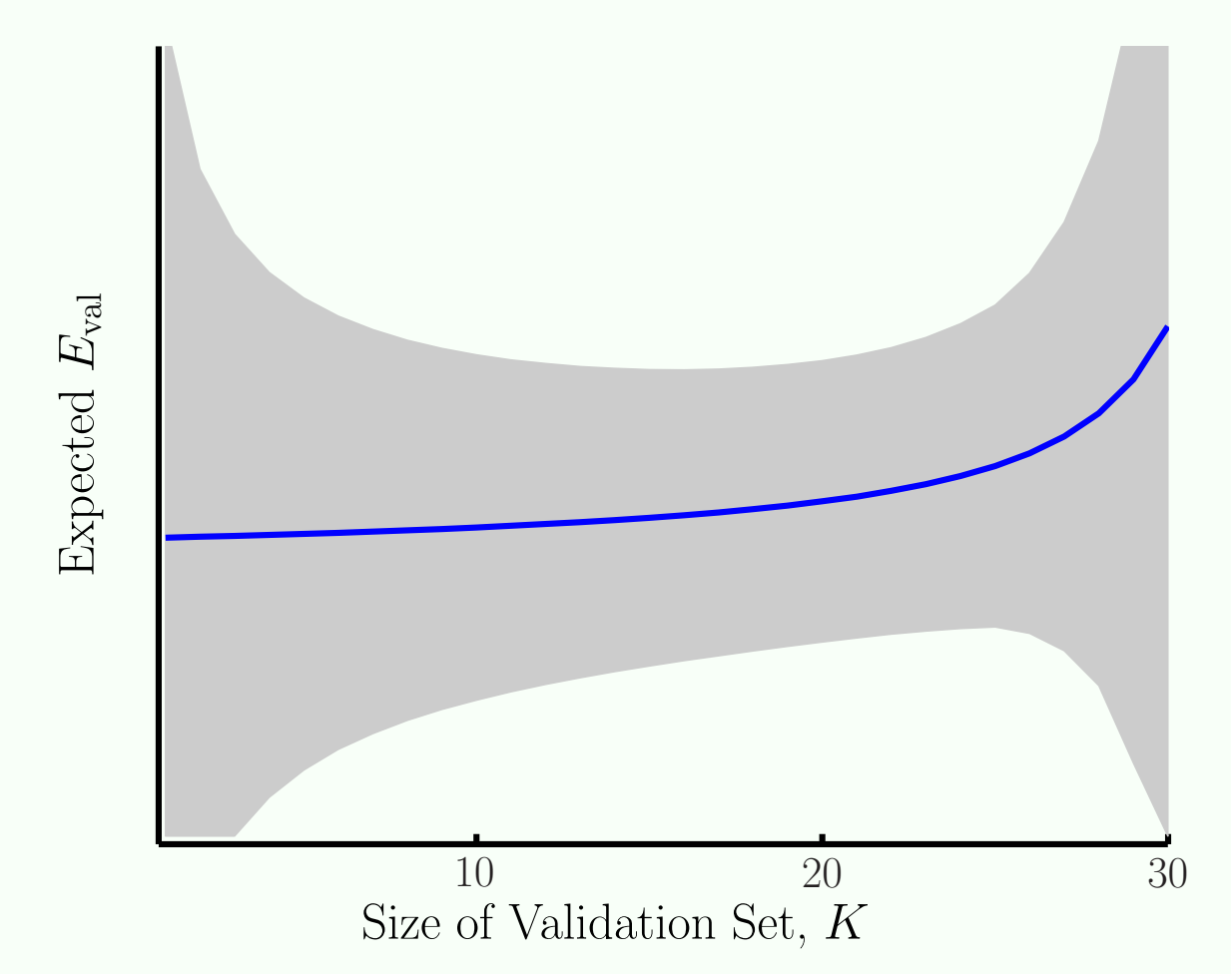
$\underbrace{K \text{ points}}_{\mathcal{D}_{\text{val}}} \rightarrow$ validation $\quad \underbrace{N - K \text{ points}}_{\mathcal{D}_{\text{train}}} \rightarrow$ training

$O\left(\frac{1}{\sqrt{K}}\right)$: Small $K \implies$ bad estimate

Large $K \implies$?

Rule of thumb: use 20% of the data for validation

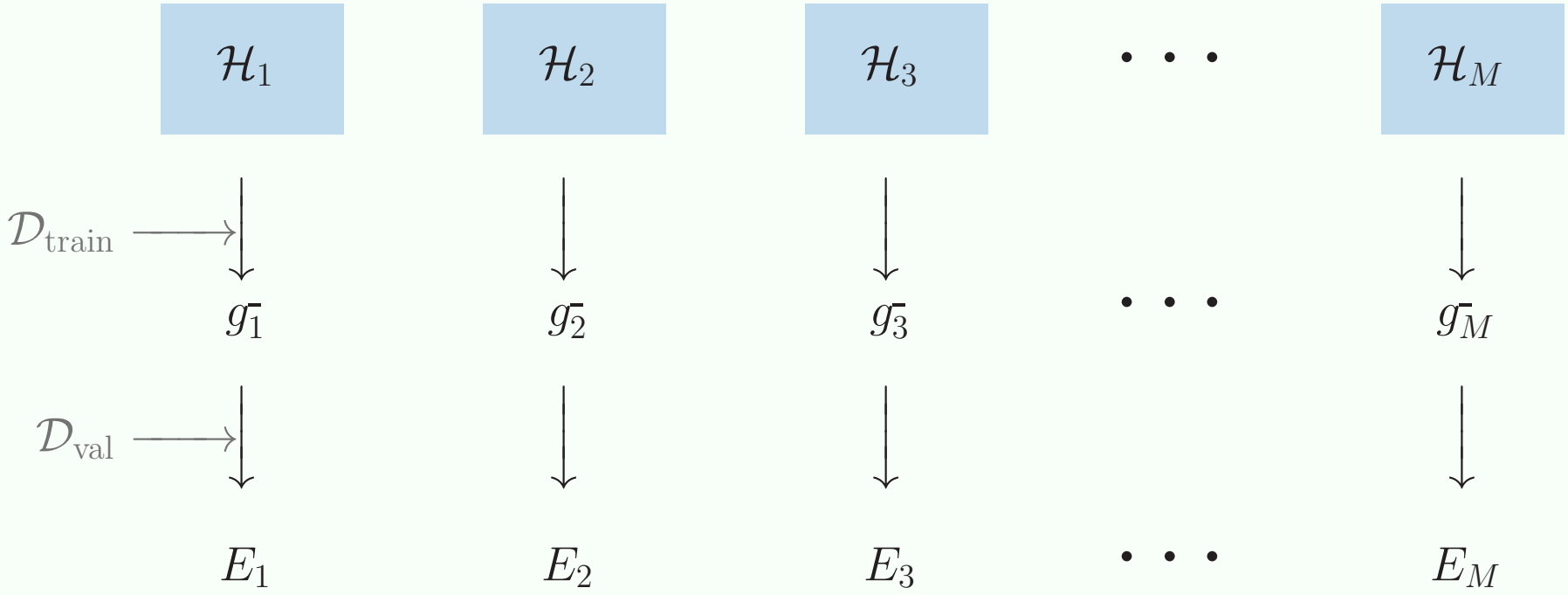
Choosing the size of the validation set



When the validation set is large, the estimate goes up because of a small training set

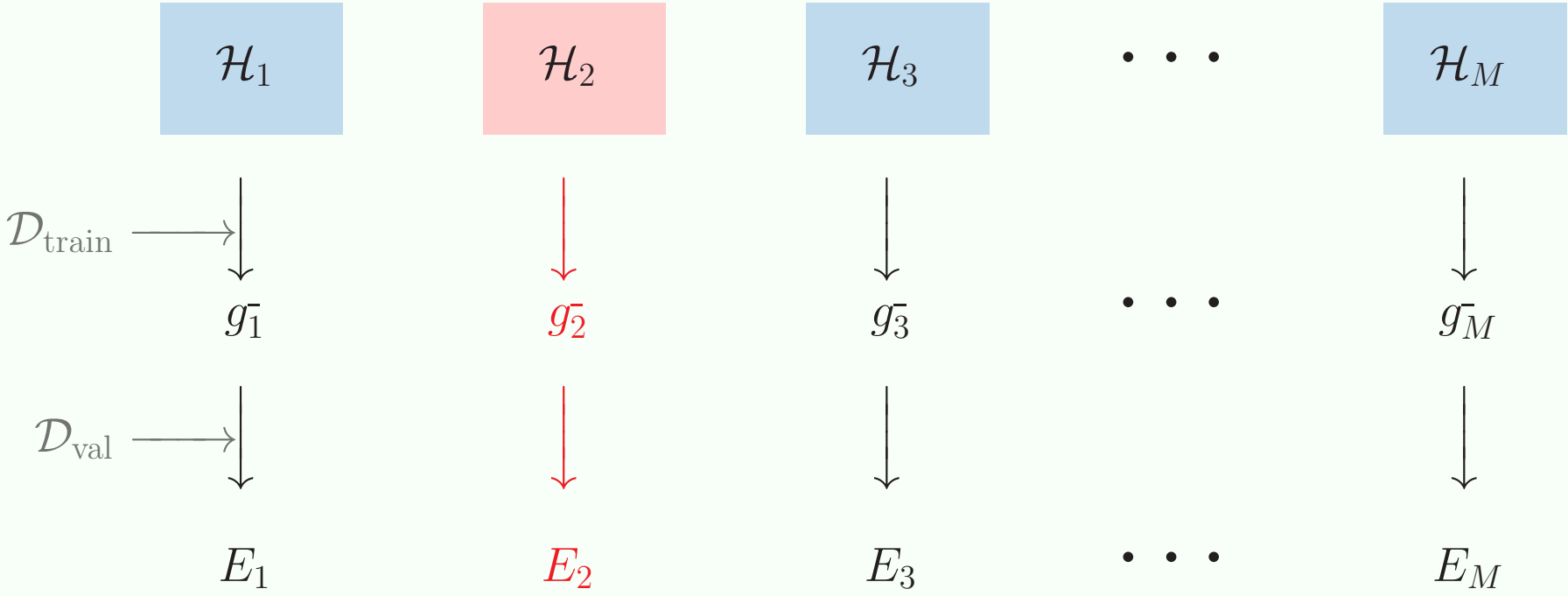
Using the validation set

The validation set is used to get estimates that allow us to choose a value for the regularization parameter.



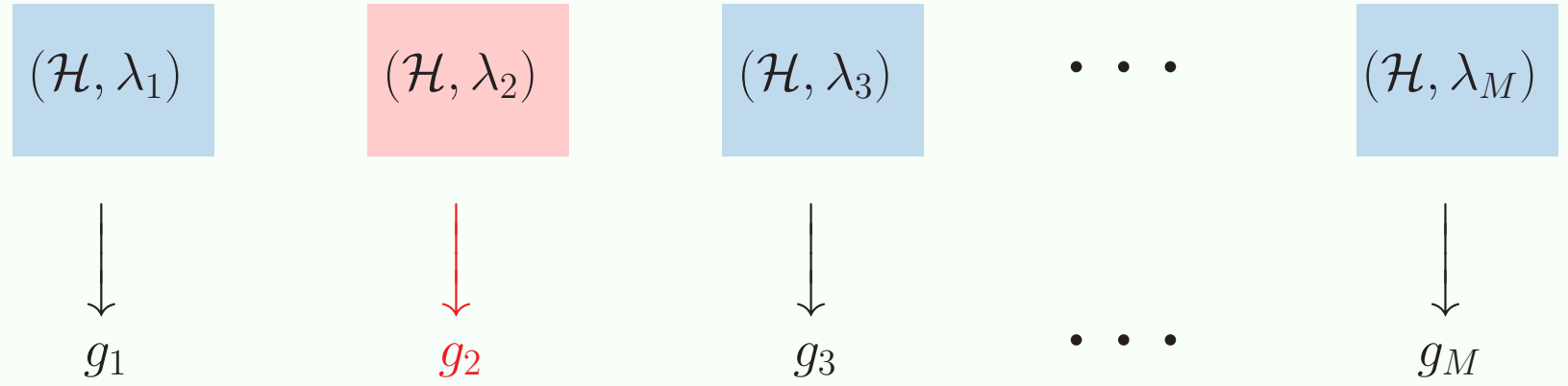
Using the validation set

The validation set is used to get estimates that allow us to choose a value for the regularization parameter.



Using the validation set

The validation set is used to get estimates that allow us to choose a value for the regularization parameter.



Using the validation set

M models $\mathcal{H}_1, \ldots, \mathcal{H}_M$

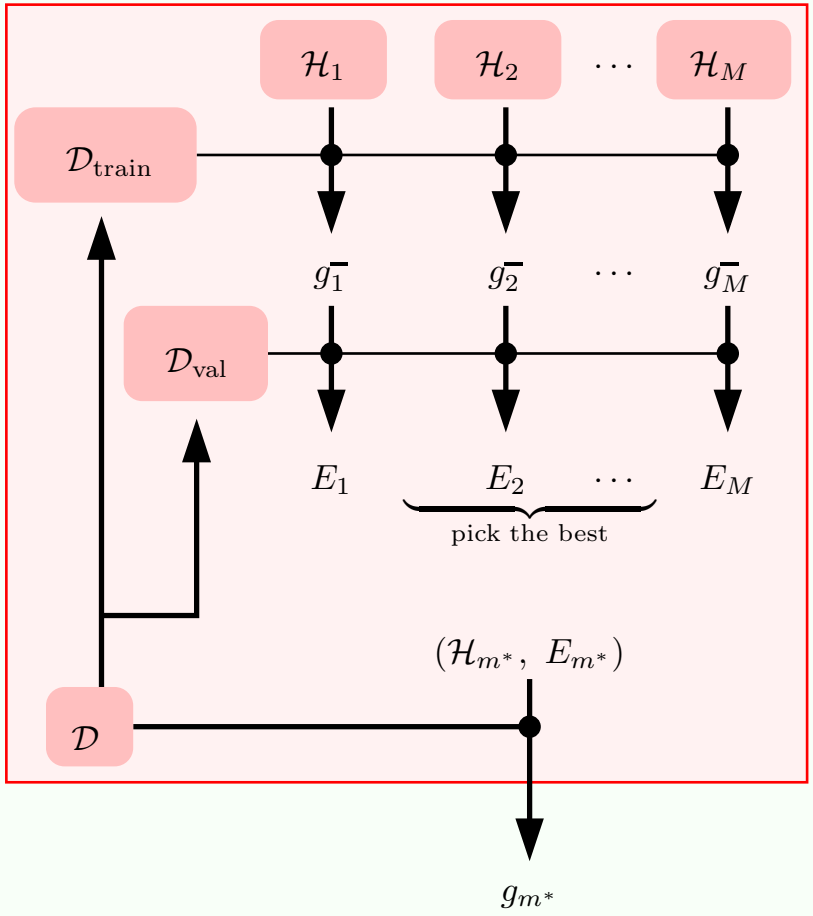
Use $\mathcal{D}_{\text{train}}$ to learn g_m^- for each model

Evaluate g_m^- using $\mathcal{D}_{\text{val}}$:

$$E_m = E_{\text{val}}(g_m^-); \quad m = 1, \ldots, M$$

Pick model $m = m^*$ with smallest E_m

At the end: train a model on all the data using the parameters of H_{m^*}.

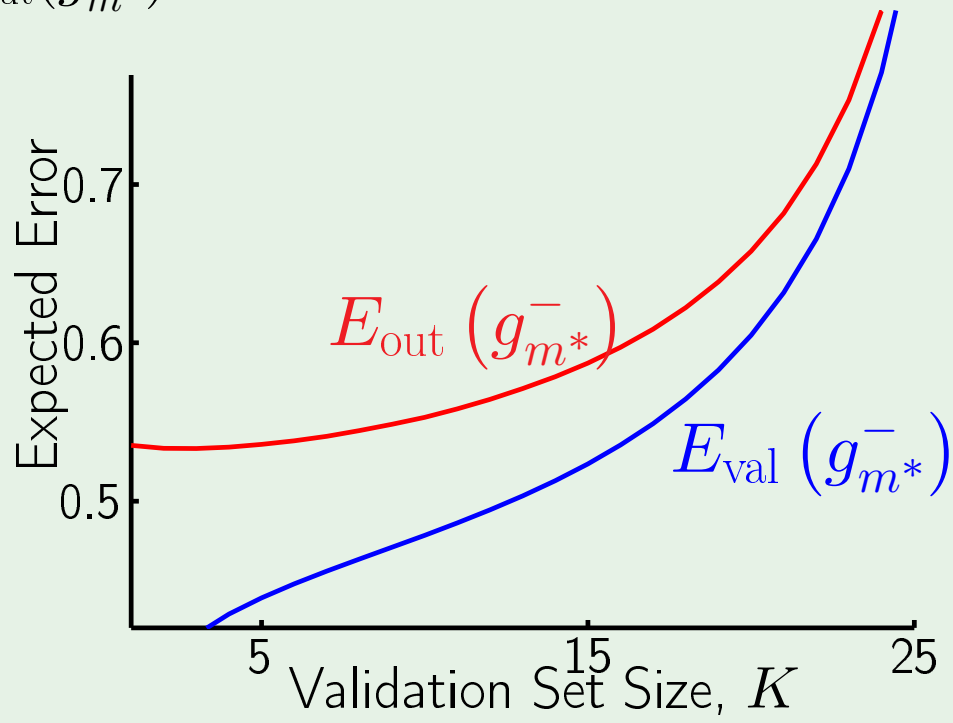


Bias

The error estimates using the validation set optimistic estimates of E_{out}!

We selected the model $\mathcal{H}_{m*}$ using $\mathcal{D}_{\text{val}}$

$E_{\text{val}}(g_{m*}^-)$ is a biased estimate of $E_{\text{out}}(g_{m*}^-)$



Bias

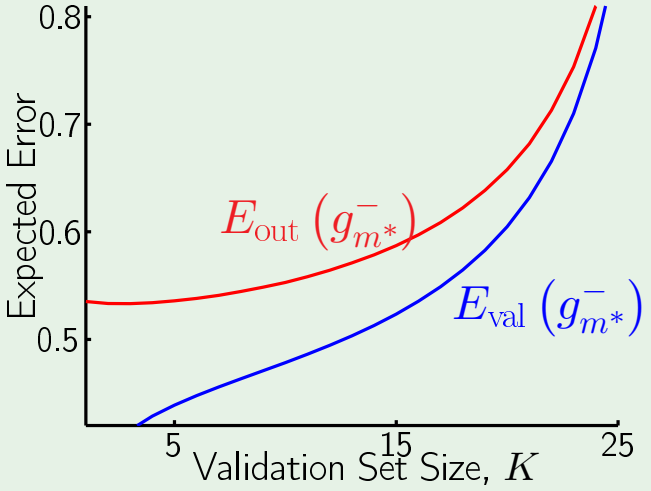
The error estimates using the validation set optimistic estimates of E_{out}!

We selected the model $\mathcal{H}_{m^*}$ using $\mathcal{D}_{\text{val}}$

$E_{\text{val}}(g_{m^*}^-)$ is a biased estimate of $E_{\text{out}}(g_{m^*}^-)$

So you need to have a separate test set.

Training set: totally contaminated
Validation set: slightly contaminated
Test set: "clean"



We have a dilemma...

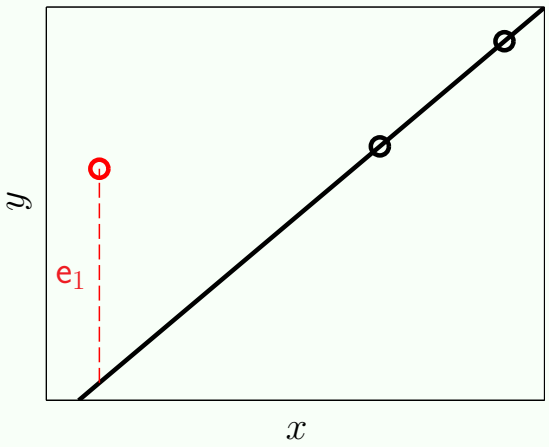
$$E_{\text{out}}(g) \approx E_{\text{out}}(g^-) \approx E_{\text{val}}(g^-)$$

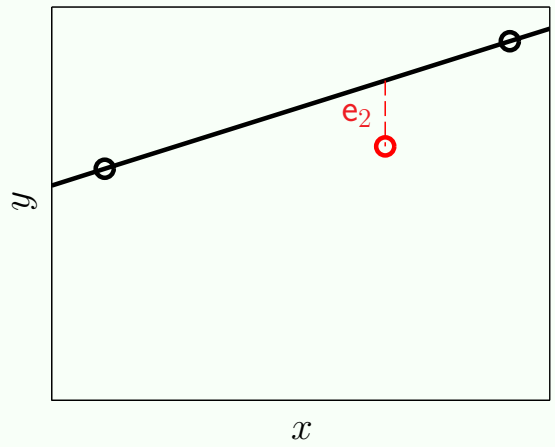
(small K) (large K)

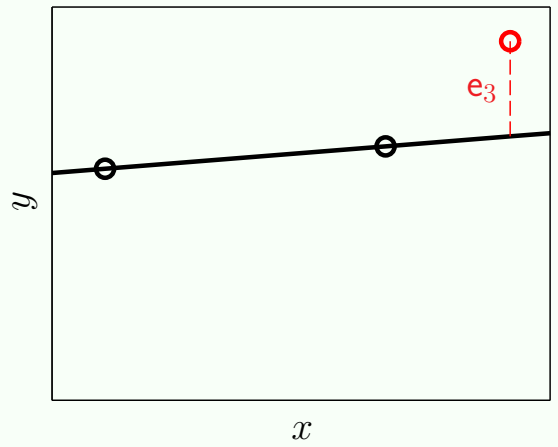
Can we have K both large and small?

Leave-one-out errors

Extreme case: K=1

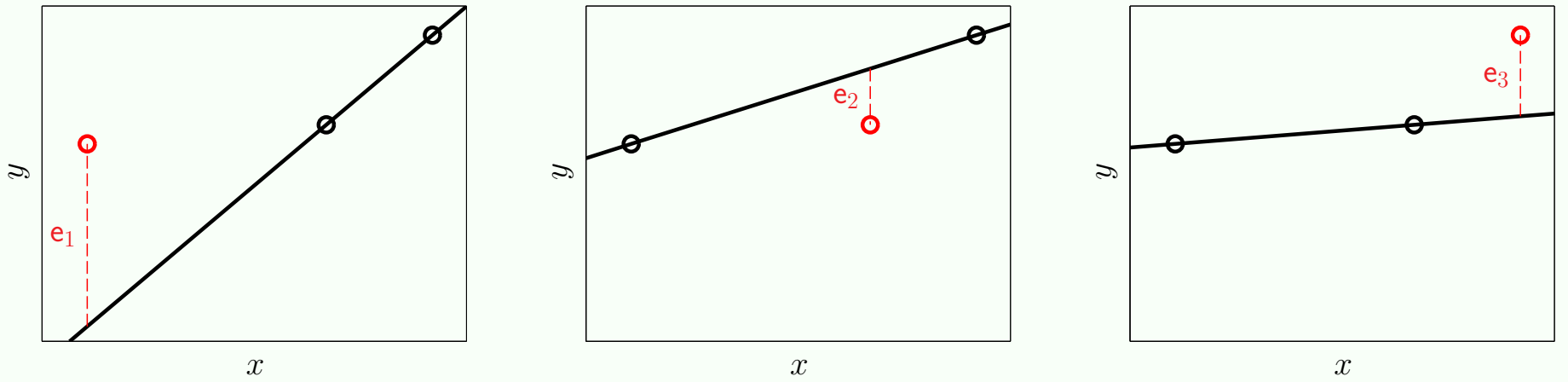






The leave-one-out estimate

Extreme case: K=1



$$\mathbb{E}[\mathsf{e}_1] = E_{\text{out}}(g_1^-)$$

$$E_{\text{cv}} = \frac{1}{N}\sum_{n=1}^{N} \mathsf{e}_n$$

Theorem. E_{cv} is an unbiased estimate of $\bar{E}_{\text{out}}(N-1)$.

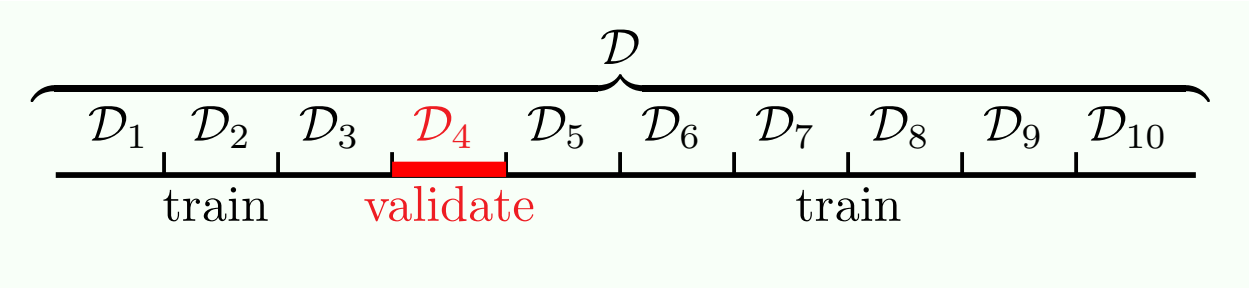
Expected E_{out} when learning with $N-1$ points.

Cross validation

The leave-one-out estimate is expensive to compute!

Cross validation:

- Randomly partition the data into k parts ("folds").
- Set one fold aside for evaluation and train a model on the remaining k-1 folds and evaluate it on the held-out fold.
- Repeat until each fold has been used for evaluation

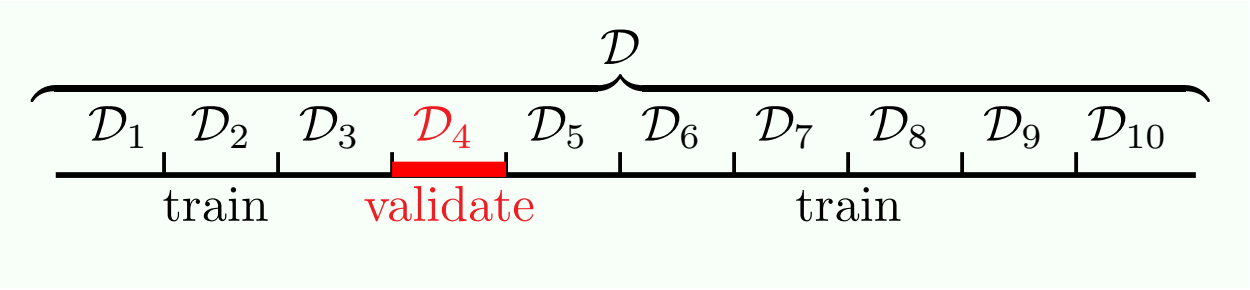


Cross validation

The leave-one-out estimate is expensive to compute!

Cross validation:

- Randomly partition the data into k parts ("folds").
- Set one fold aside for evaluation and train a model on the remaining k-1 folds and evaluate it on the held-out fold.
- Repeat until each fold has been used for evaluation



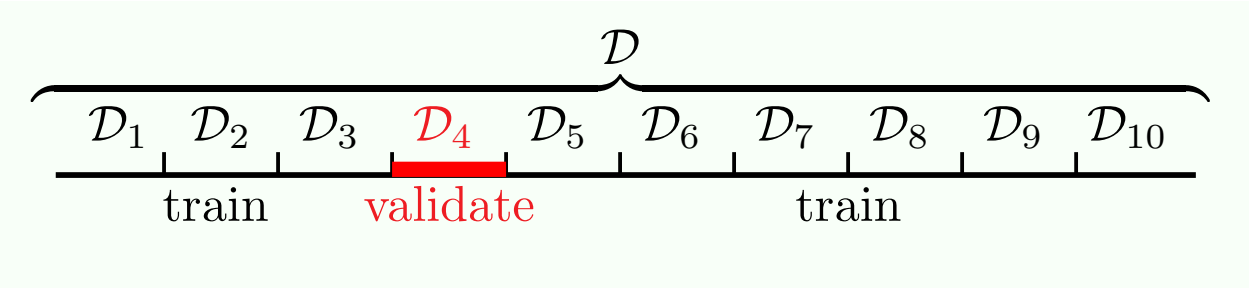
- The reported error is the average over the errors for each fold.

Cross validation

The leave-one-out estimate is expensive to compute!

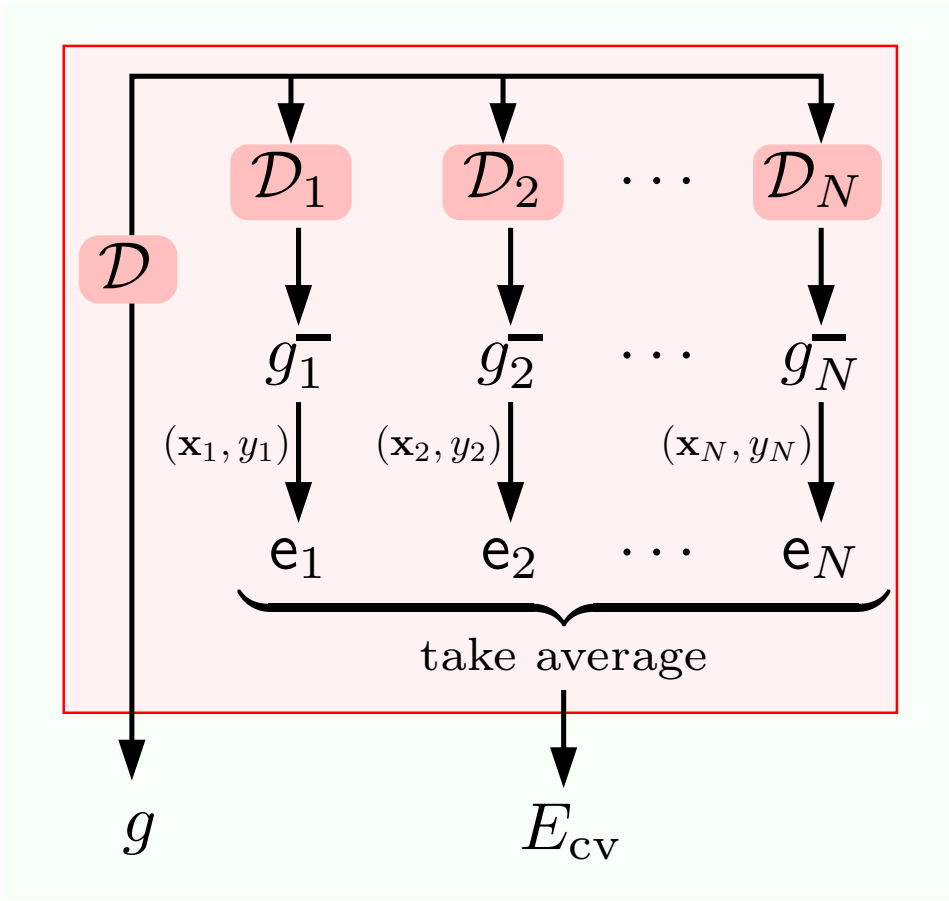
Cross validation:

- Randomly partition the data into k parts ("folds").
- Set one fold aside for evaluation and train a model on the remaining k-1 folds and evaluate it on the held-out fold.
- Repeat until each fold has been used for evaluation



Stratified-cross validation aims at achieving roughly the same class distribution in each fold.

Using cross-validation



customer

select the best model

Measures of classifier performance

Classifier performance can be summarized by a table known as the confusion matrix or contingency table:

true labels	predicted labels:	
	-1	1
-1	1439	61
1	62	1438

Measures of classifier performance

Let's take a closer look at the contingency table:

true labels	predicted labels:	
	-1	1
-1	1439	61
1	62	1438

How do we compute error from the contingency table?

Measures of classifier performance

For binary classification problems it is customary to express the contingency table as:

true labels	predicted labels: -1	predicted labels: 1
-1	TN	FP
1	FN	TP

TP – number of true positives

TN – number of true negatives

FP – number of false positives

FN – number of false negatives

Measures of classifier performance

For binary classification problems it is customary to express the contingency table as:

true labels	predicted labels:		
	-1	1	
-1	TN	FP	Neg = TN+FP
1	FN	TP	Pos = TP+FN

True positive rate/sensitivity/recall: TP / Pos

True negative rate/specificity: TN / Neg

False positive rate: FP / Neg

Precision: TP / (TP + FP)

Measures of classifier performance

Suppose you have a dataset with very few positive examples compared to negative examples (unbalanced data)

A classifier that classifies every example as negative would still attain high accuracy (this is called the majority class classifier).

Need an alternative measure of accuracy!

The choice of classification threshold

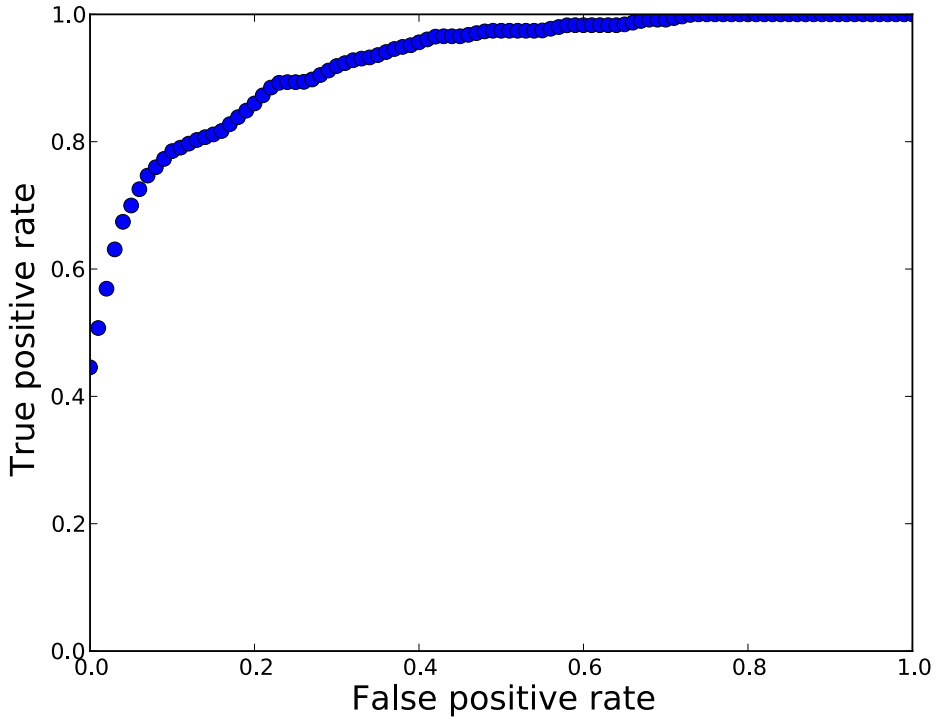
All the classifiers we will study provide a scoring function whose magnitude indicates how sure we are it belongs to a given class. For example: $\mathbf{w}^\mathsf{T}\mathbf{x} + b$

The choice of the threshold is somewhat arbitrary, and in a given application we may prefer to ignore positive predictions that are associated with small scores

To have a view of classifier performance that is independent of the choice of threshold we consider the ROC curve.

ROC curve

The ROC curve is a plot of the true positive rate as a function of false positive rate as you vary the classification threshold



How does the ROC curve of a perfect classifier look like?

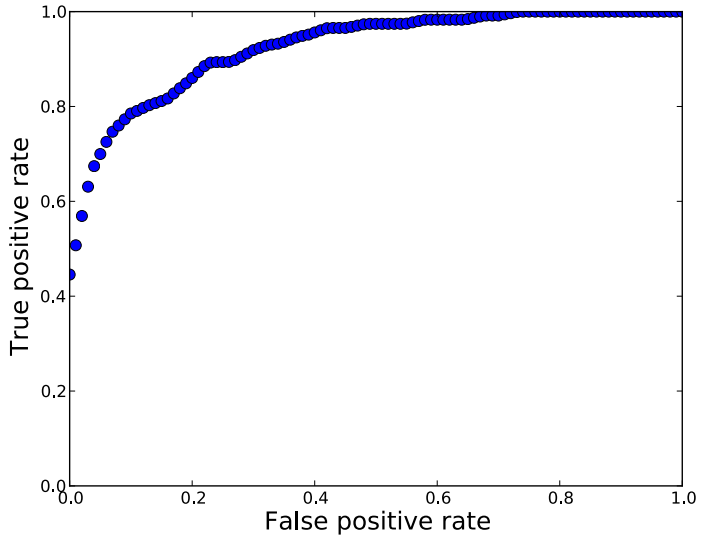
For a random classifier?

ROC curve computed on the heart disease dataset from the UCI repository

ROC curves and ranking

An ROC curve is often summarized by the area under the curve (AUC).

AUC = 0.92



AUC is essentially the probability that a positive example will get a higher score than a negative example

ROC curves

This is also a nice way of comparing classifiers: