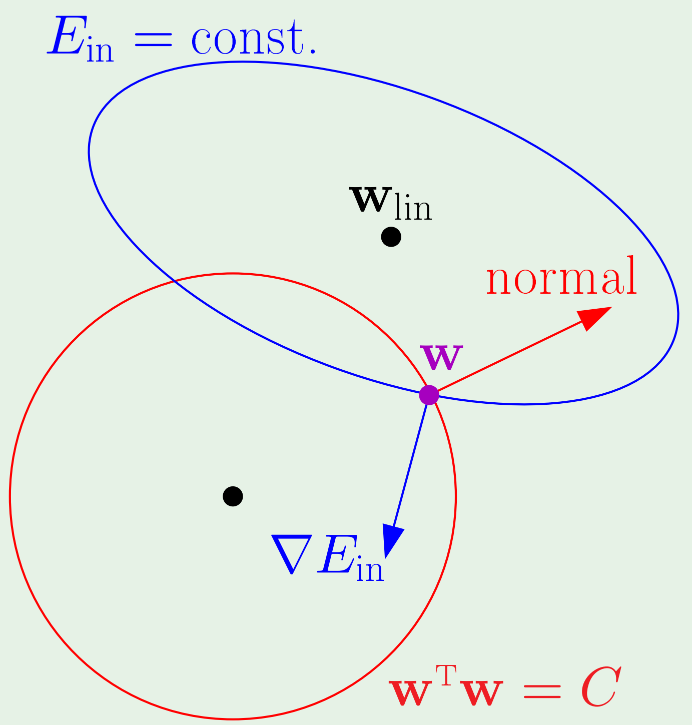
Review of Lecture 12

- Regularization

constrained $\longrightarrow$ unconstrained

(forbid some h from being considered, so reduce VC dim/smooth the function, so better generalization)



$E_{\text{in}} = \text{const.}$
$\mathbf{w}_{\text{lin}}$
normal
$\mathbf{w}$
∇E_{in}
$\mathbf{w}^{\mathsf{T}}\mathbf{w} = C$

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

create unconstrained version, no specific h is prohibited, but we have a preference of weights based on a penalty (which is related to the constraint)

- Choosing a regularizer

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

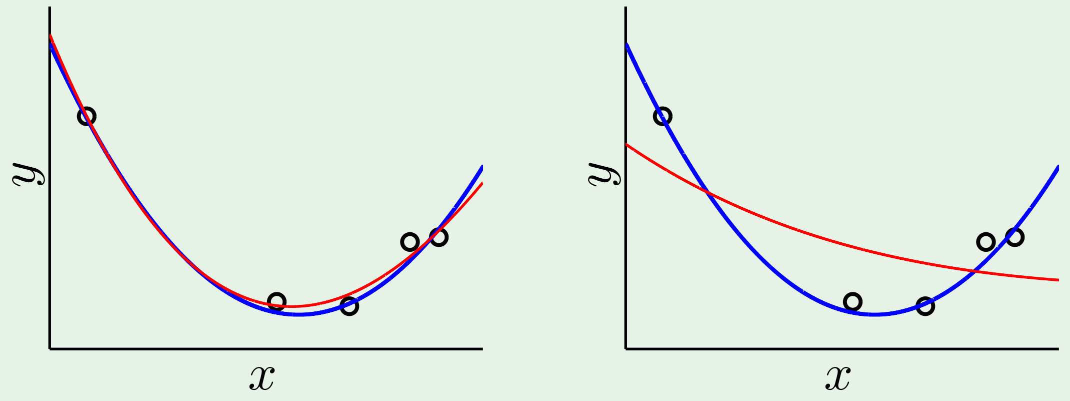
$\Omega(h)$: is a heuristic choice $\longrightarrow$ smooth, simple h

most used: **weight decay**

Eaug is a better proxy for Eout than Ein, it is a better quantity to minimise to minimise Eout

λ: principled; validation

$\lambda = 0.0001$ $\lambda = 1.0$



Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 13: Validation

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Tuesday, May 15, 2012

Outline

- The validation set

- Model selection

- Cross validation

Validation versus regularization

In one form or another, $E_{\text{out}}(h) = E_{\text{in}}(h) + \text{overfit penalty}$

Regularization:

$$E_{\text{out}}(h) = E_{\text{in}}(h) + \underbrace{\text{overfit penalty}}_{\text{regularization estimates this quantity}}$$

Validation:

$$\underbrace{E_{\text{out}}(h)}_{\text{validation estimates this quantity}} = E_{\text{in}}(h) + \text{overfit penalty}$$

Analyzing the estimate

On out-of-sample point $(\mathbf{x}, y)$, the error is $\mathbf{e}(h(\mathbf{x}), y)$

Squared error: $\left(h(\mathbf{x}) - y\right)^2$

Binary error: $[\![h(\mathbf{x}) \neq y]\!]$

Expected value with respect to the choice of k (see next slide), with the probability distribution over the input space that generates x.

$$\mathbb{E}\left[\mathbf{e}(h(\mathbf{x}), y)\right] = E_{\text{out}}(h)$$

(so the estimate is not biased - it is as likely to be optimistic as it is pessimistic, although it is likely to be poor since it only depends on one point, so variance is likely to be large - hence use a full set (see next slide))

$$\text{var}\left[\mathbf{e}(h(\mathbf{x}), y)\right] = \sigma^2$$

From a point to a set

(out-of-sample, so points not used in training)

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

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

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

K^2 total terms in the variance expression. The off-diagonal cross terms for different k, which represent the covariance between the errors on different points, are all zero since the points are picked independently. We therefore only have the diagonal elements = sigma^2. So larger K shrinks the error bar so Eval can be a reliable estimate of Eout.

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

assuming sigma is constant in the range we are using, so no dependency of it on K

K is taken out of N

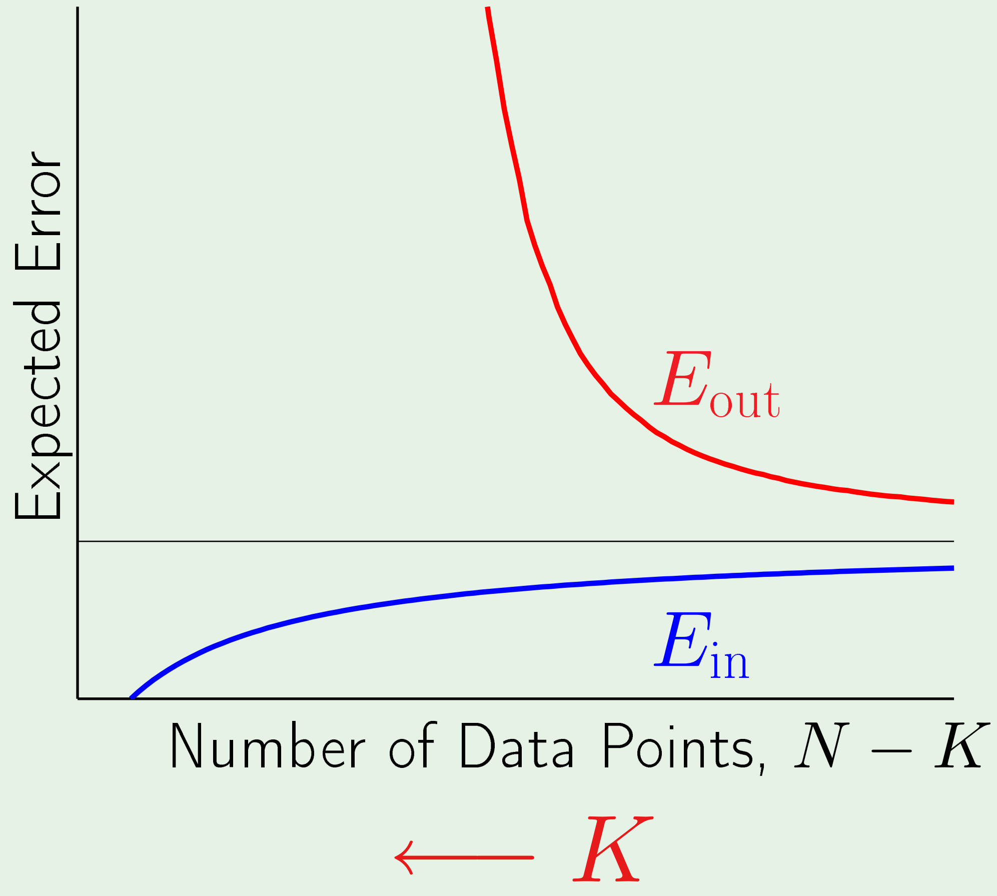
(How many data from D should we use to train and how many to validate)

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

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

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

Large $K \implies$?



This graph suggests that increasing K decreases Ein and Eout, so we get a more reliable estimate of a worse quantity since using fewer points for training means the model fits the noise. So, can we use K to estimate the error, then restore the dataset and train on the full set so we get a better model - see next slide.

K is put back into N

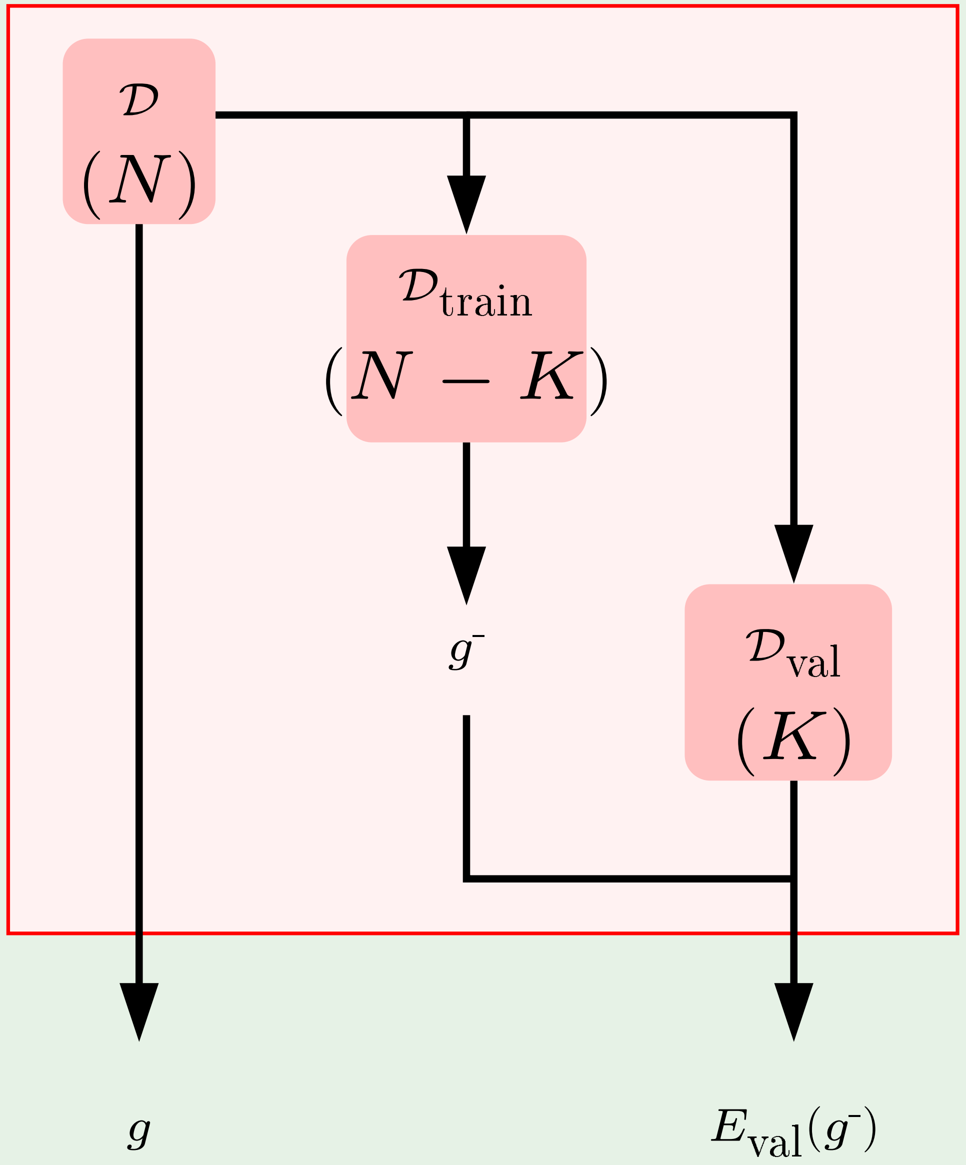
$$\begin{tikzcd}
\mathcal{D} \arrow[r] \arrow[d] & \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}} \\
N & N-K \quad K
\end{tikzcd}$$

$$\mathcal{D} \Longrightarrow g \qquad \mathcal{D}_{\text{train}} \Longrightarrow g^-$$

$E_{\text{val}} = E_{\text{val}}(g^-)$ Large $K \Longrightarrow$ bad estimate!

Rule of Thumb:

$$K = \frac{N}{5}$$



$\mathcal{D}$ (N)
$\mathcal{D}_{\text{train}}$ $(N-K)$
g^-
$\mathcal{D}_{\text{val}}$ (K)
g
$E_{\text{val}}(g^-)$

So here we report g as our final (best) hypotheses, but Eval(g-) is the validation error of a different hypothesis. If the difference between Eval(g-) and Eval(g) is large, the estimate is bad. This what happens when we have large K: the difference between g and g- is larger, so the estimate on g- will be a poor estimate for g, so we have a bad estimate again. Meanwhile for small K, the Eval is not reliable (it is a bad estimate) since the variance is big. Hence we need a compromise for K: not too small such as to have large fluctuations in the estimate, and also not too big so that the estimate is not too far from that of the hypothesis we are reporting.

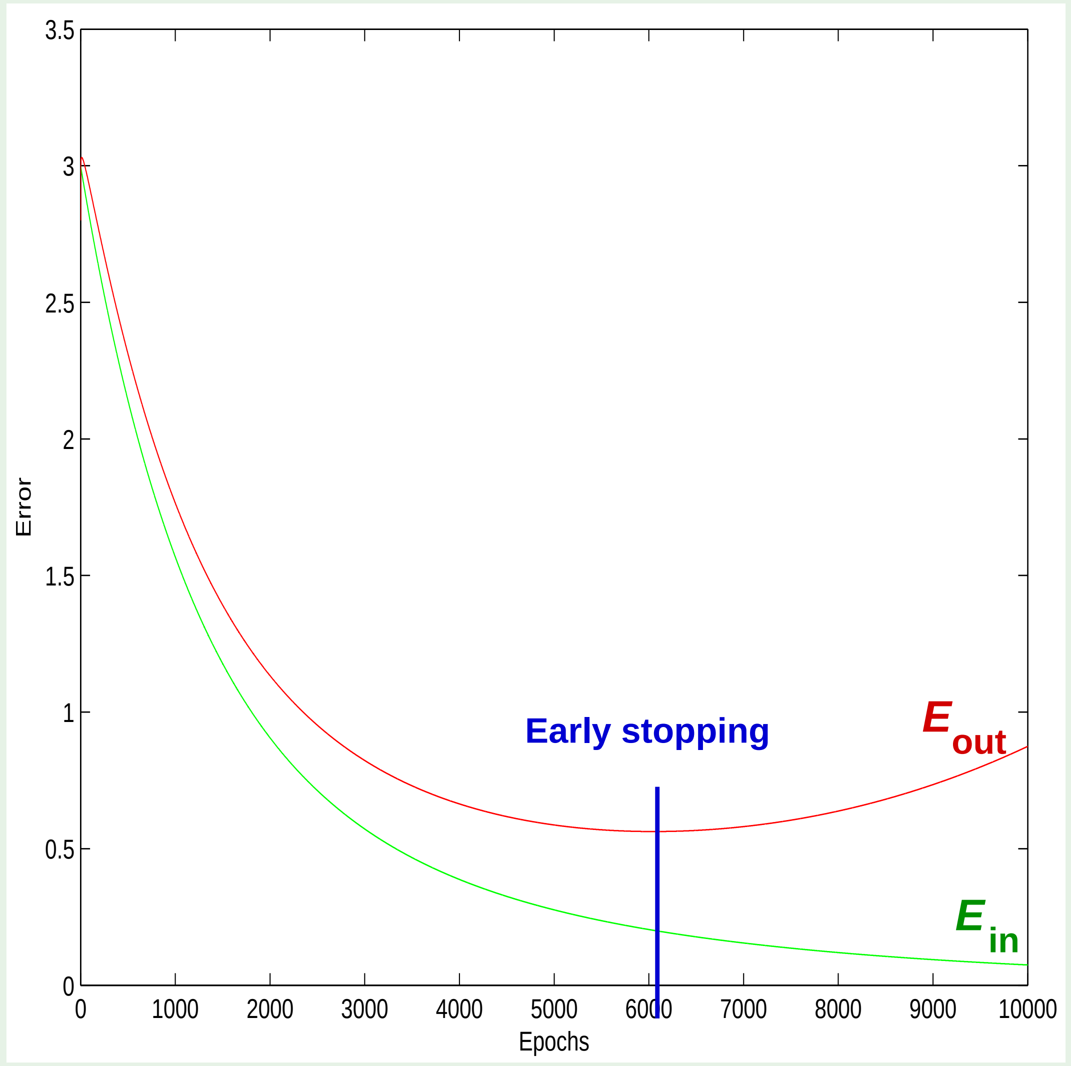
Why 'validation'

$\mathcal{D}_{\text{val}}$ is used to make learning choices

If an estimate of E_{out} affects learning:

the set is no longer a **test** set!

It becomes a validation set



What's the difference?

Test set is unbiased; validation set has optimistic bias (we deceive ourselves - the optimistic bias is always in the direction of thinking that Eout is smaller than it will turn out to be)

Example:

Two hypotheses h_1 and h_2 with $E_{\text{out}}(h_1) = E_{\text{out}}(h_2) = 0.5$

Error estimates $\mathbf{e}_1$ and $\mathbf{e}_2$ uniform on $[0, 1]$

(somewhere in the range [0, 1])

Pick $h \in \{h_1, h_2\}$ with $\mathbf{e} = \min(\mathbf{e}_1, \mathbf{e}_2)$

Now the measurements of the error impact the choice of h

$\mathbb{E}(\mathbf{e}) < 0.5$ optimistic bias

With two variables e1 and e2, the probability the minimum of the two is less than 0.5 is 75% (this is because there a four possiblities of where e1 and e2 can lie: e1 < 0.5 and e2 not, e2 < 0.5 and e1 not, e1 & e2 < 0.5, or e1 & e2 > 0.5. SInce three of the four involve the minimum being < 0.5, the prob is 75%). Hence this is an optimistic bias. Fortunately, this utility of validation in machine learning is so light (choice of a parameter like lambda, or model selection) that we can 'swallow' the bias - it is minor and we will not estimate many quantitites and keep applying bias such that Eval becomes training error in disguise. With a respectable size of validation set, we get a pretty reliable estimate of Eout.

Outline

- The validation set

- Model selection

- Cross validation

- the main use of validation sets (the choice of lambda happens to be a manifestation of this)

Includes selection between linear models, NN, SVM etc., or between polynomial models 2nd, 5th, 10th order, or between values of lambda 0.01, 0.1 or 1 for 5th order polynomials. In other words, model selection is used whenever a choice has to be made and we want to make it in a principled way based on Eout (since this is the bottom line), and we are going to use the validation set to do that.

Using $\mathcal{D}_{\text{val}}$ more than once

(see last slide on the variety of what H can represent, it represents models in a general sense)

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

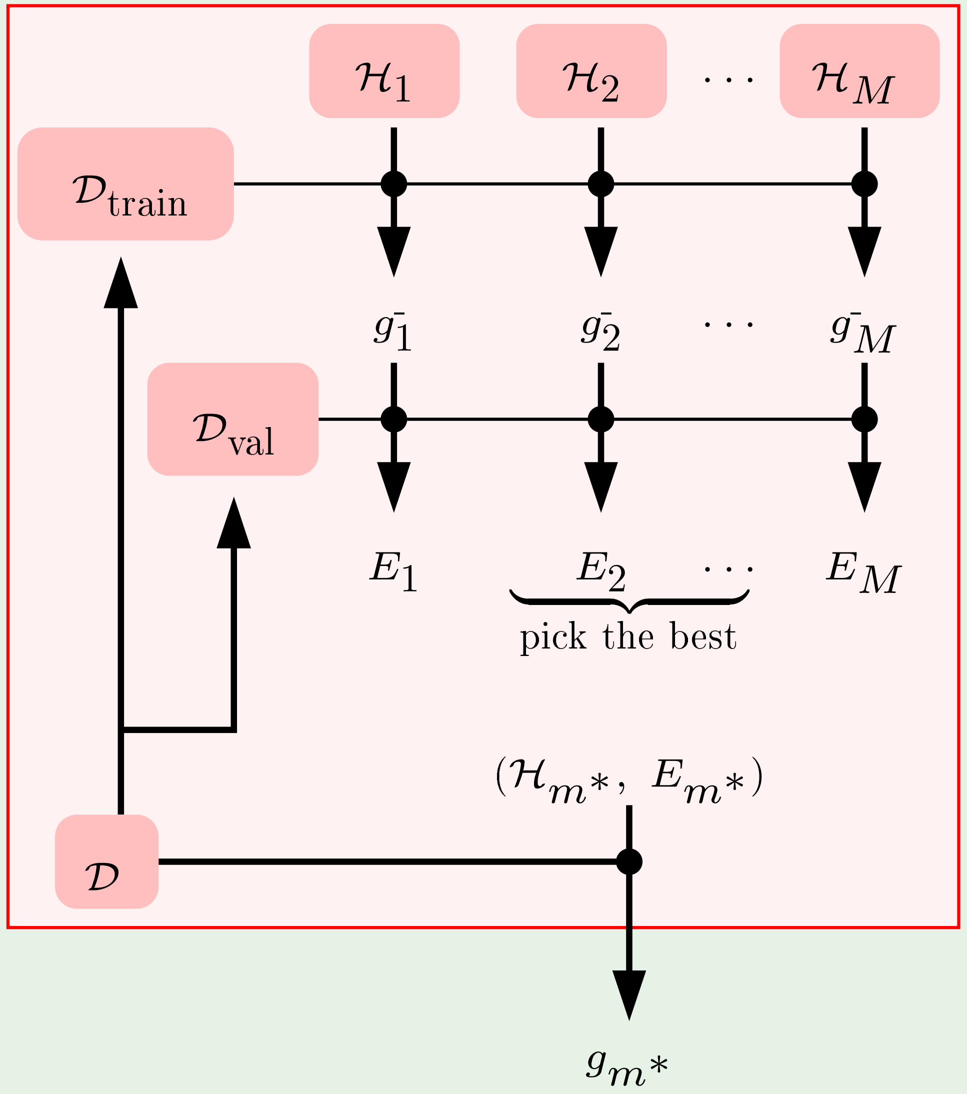
Use $\mathcal{D}_{\text{train}}$ to learn g_m^- for each model

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

Each Em is an unbiased estimate of Eout for the corresponding hypothesis; we pick the smallest of them, so a bias is introduced. When we use the estimate to choose, the estimate is no longer reliable since we particularly chose for it, so now it looks optimistic because by choice it has good performance - not because it has an inherently good performance but because we looked for the model with the good performance



The bias

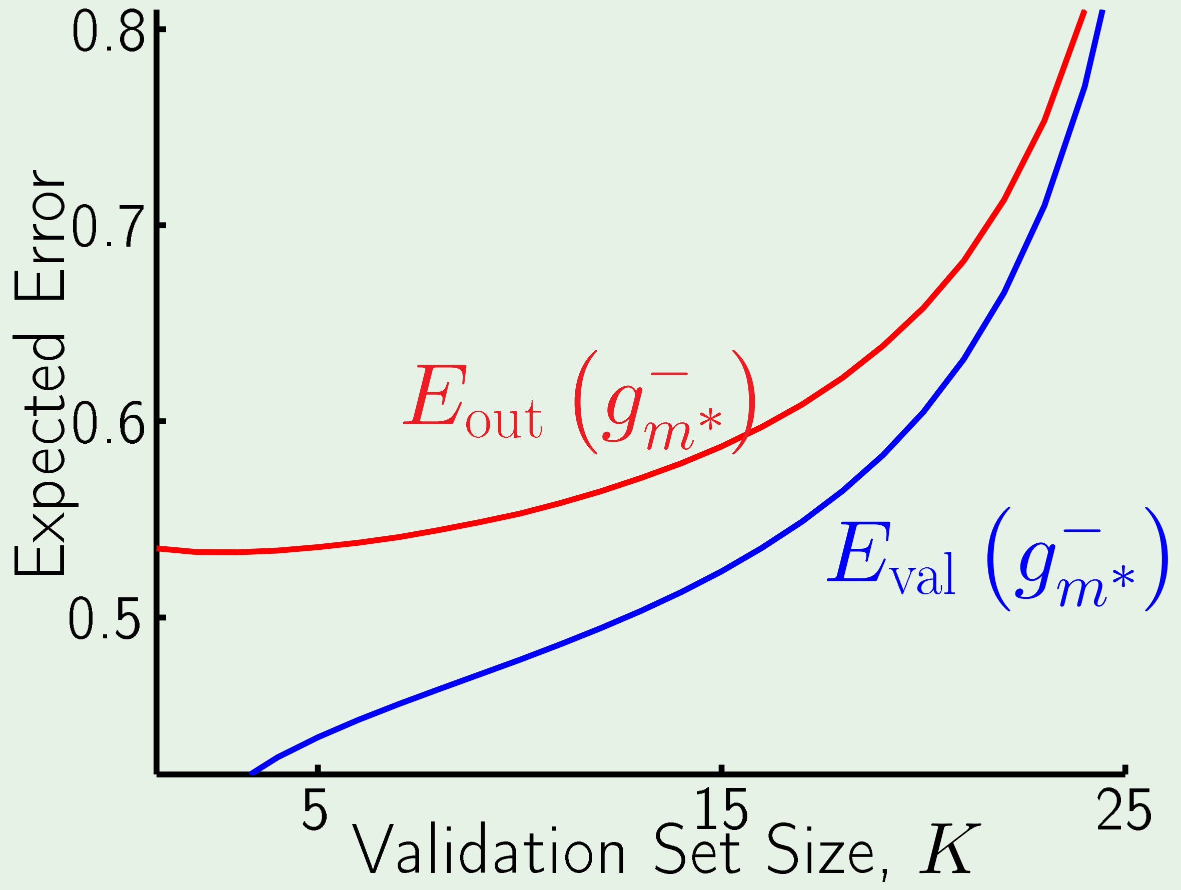
We selected the model $\mathcal{H}_{m^*}$ using $\mathcal{D}_{\text{val}}$

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

Illustration: selecting between 2 models

Experiment: we choose between two models, 2nd or 5th order polynomials. Over a large number of runs we make a choice between the models based on the validation set. After that, we look at the actual Eout to see if there is a systematic bias in the one we choose with respect to its out of sample error. In each run, we may have chosen H2 or H5, whichever gives the smallest Eval, and we take an average of Eval and Eout over all the runs for a given K. The total number of data points is around 30.

The curves go up because as K increases, N-K decreases so fewer examples to train on to get g-, so worse Ein - therefore the graph is like a reversed learning curve. The curves get closer to each other as K increases because the estimate is more and more reliable, so Eval better approximates Eout.



How much bias

We are using the validation set to estimate Eout and we are claiming it close estimate; we realize if we don't use it too much we are OK, but what is a guideline for too much?

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

$\mathcal{D}_{\text{val}}$ is used for "training" on the **finalists model**:

$$\mathcal{H}_{\text{val}} = \{g_1^-, g_2^-, \ldots, g_M^-\}$$

Using a learning algorithm we train H1,...,HM on the training set D_train to get the 'finalists' g1-,...,gM-, so the hypothesis set we 'train' on now is Hval - as far as the validation set is concerned it does not know what happened before. All we do it give it the finalists 'hypothesis set' and ask for Dval to choose the one with the minimum error - this is equivalent to training. Since Eval is really the 'training error' on this special set, we can use Hoeffding and VC:

Back to Hoeffding and VC!

Ein + penalty for model complexity

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

(even when using the simple union bound)

regularization λ early-stopping T

early-stopping T represents the number of epochs at which we choose to stop to minimize Eout, preventing/reducing overfitting

When there is a choice between a continuous set/infinite number of models, from VC analysis we use the effective complexity (VC dimension) of what we are doing. So say we are choosing lambda analytically and therefore we allow the numeric value to be whatever is best for regularization, we simply look at it as a problem witha single d.o.f., so VC dimension 1. In the case we have a few parameters to choose, if we have a reasonable size of K (instead of reasonable N in VC analysis), then we can be confident that Eval will not be far from Eout. In the cases, like lambda and T, where there is a continuity to it and only one parameter is being chosen, essentially corresponds to one d.o.f. So we use K=100 to choose 2 parameters, we are OK; however, choosing 20 parameters with K=100 means the estimate will be ruined - we are now actually training. There is a grey area between validation and training and if we push our luck our validation estimate loses its main attraction which is that it is a reasonable estimate of Eout and the reliability goes down.

Data contamination

Error estimates: $E_{\text{in}}, \ E_{\text{test}}, \ E_{\text{val}}$

Using the data to make choices or decisions contaminates it as far as its ability to estimate the real performance is concerned.

Contamination: Optimistic (deceptive) bias in estimating E_{out}

Training set: totally contaminated

Important to keep the validation set only slightly contaminated (i.e. only making a few choices) - need to start with a regime where we have a number of validation sets so when one gets contaminated, we over onto another one which has not been used to make decisions and so its estimates will be reliable

Validation set: slightly contaminated

Test set: totally 'clean'

Test set gives an unbiased estimate, so the customer is as likely to be pleasantly surprised as unpleasantly surprised - if the test set is large, they are likely not surprised at all since Etest will be very close to Eout

Outline

- The validation set

- Model selection

- Cross validation

The dilemma about K

We are no longer concerned with biased/unbiased, we now look for a better estimate (smaller error bars/fluctuations) of Eval for Eout

The following chain of reasoning:

$$E_{\text{out}}(g) \underset{\text{(small } K\text{)}}{\approx} E_{\text{out}}(g^-) \underset{\text{(large } K\text{)}}{\approx} E_{\text{val}}(g^-)$$

(so g- is close to g) (since for large K, Eval is a better estimate of Eout)

highlights the dilemma in selecting K:

Can we have K both small and large? ☺

Leave one out

$N-1$ points for training, and 1 point for validation!

(So g- will be close to g as N-K is close to N)

$$\mathcal{D}_n = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{n-1}, y_{n-1}), \cancel{(\mathbf{x}_n, y_n)}, (\mathbf{x}_{n+1}, y_{n+1}), \ldots, (\mathbf{x}_N, y_N)$$

Final hypothesis learned from $\mathcal{D}_n$ is g_n^-

gn- is trained on all data except for (xn,yn)

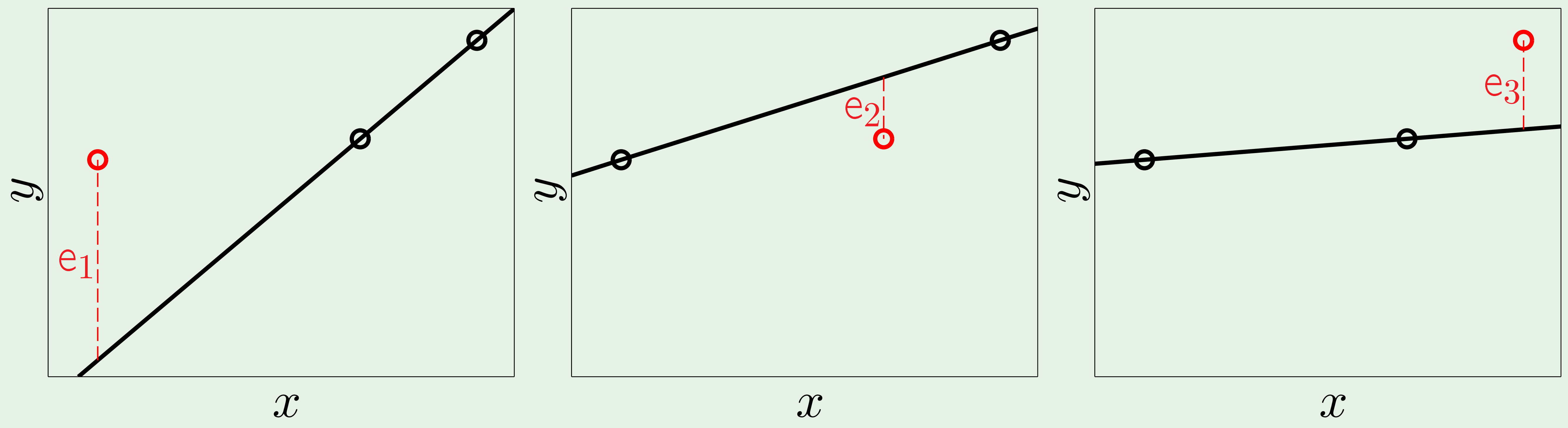
$$\mathbf{e}_n = E_{\text{val}}(g_n^-) = \mathbf{e}\left(g_n^-(\mathbf{x}_n), y_n\right)$$

cross validation error: $$E_{\text{CV}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{e}_n$$

All hypotheses will all be trained on a different set of N-1 data points. In spite of the fact these are different hypotheses, the fact they are all come from the same number of points suggests they are all realizations of something which is the expected value of all of them (from the idea of the learning curve). So e_n estimates Eval (only an esitmate since we are only validating using one data point) and Eval 'estimates' the error of the expected value on N-1 examples regardless of the identity of the examples.

So w.r.t. the dilemma, we use N-1 points to train the hypotheses (so g- is close to g) and N points to validate (so K is large). The catch is that the e_n are not independent - take e1 and e3: e1 was used to evaluate the error on a hypothesis which involved the 3rd example while e3 was used to evaluate on the 3rd example but on a hypothesis which involved the 1st example. Surprisingly the effective K is close to N (as if they were independent), doing the variance analysis, maybe effective K=95 with N=100, so very efficient.

Illustration of cross validation



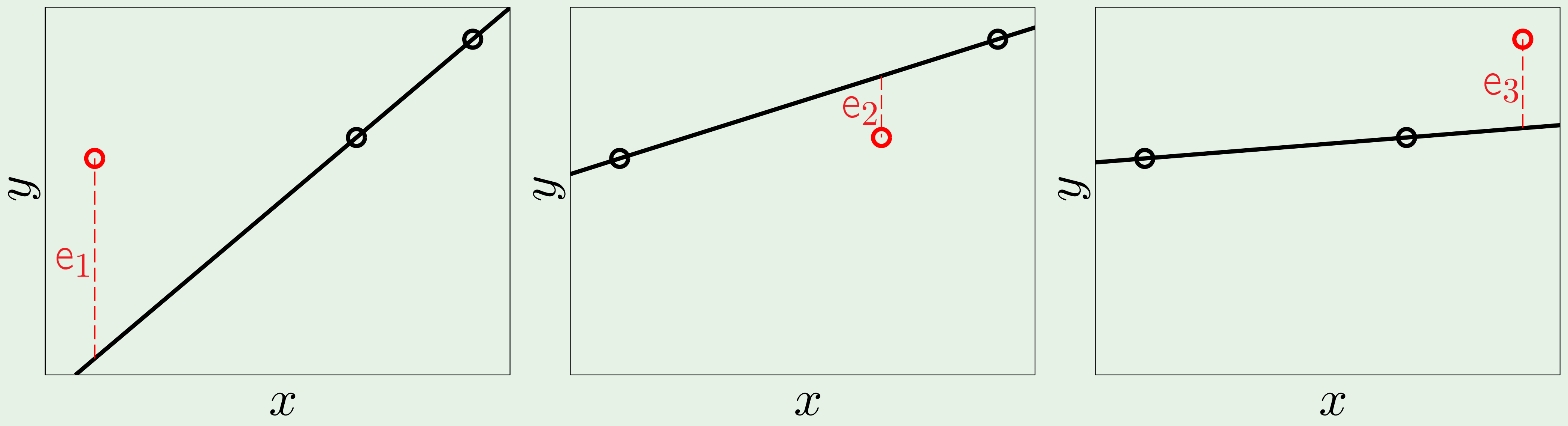
$$E_{\text{cv}} = \frac{1}{3}\left(\, e_1 + e_2 + e_3 \,\right)$$

- we take Ecv as an indication for how well the linear model fits the data out of sample

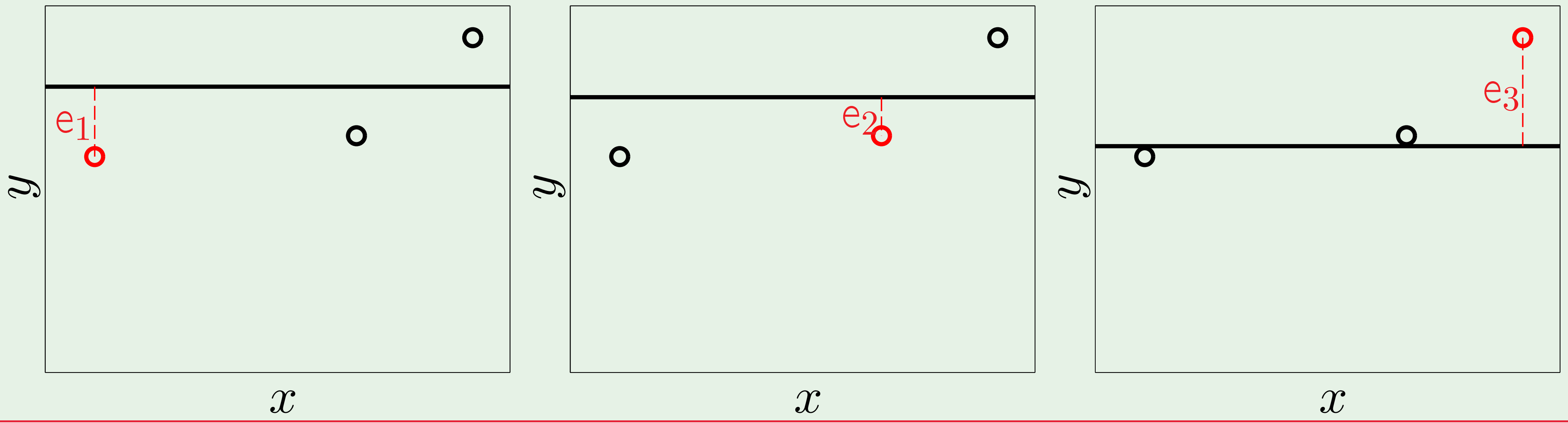
e could be squared error (for example)

Model selection using CV

Linear:



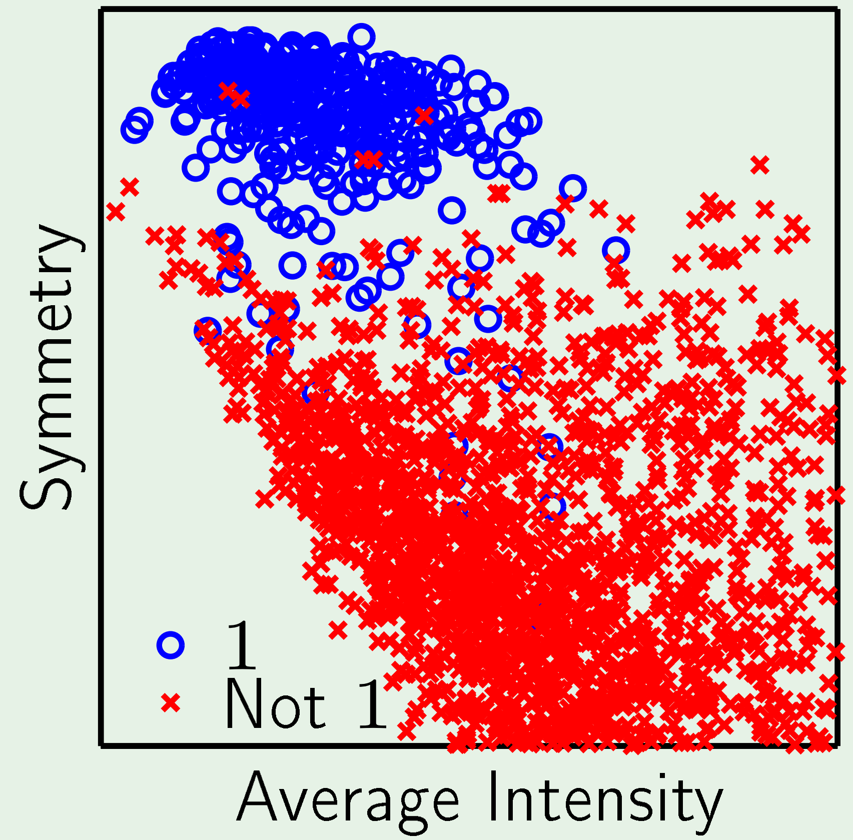
Constant:



If the question is: is the linear model better than the constant model in this case, all we have to look at is the cross validation error (it acts as a grade for each) - the constant model therefore wins. Reassuringly, the three points were generated from a constant model (+ noise), which indicates that at least in this example the cross validation error was genuinely a good indicator for selecting/ranking models. On average, cross-validation gives us the correct decision. This systematic, principled method also avoids using funny heuristics that could be applied; remember - you can fool yourself into any pattern you want.

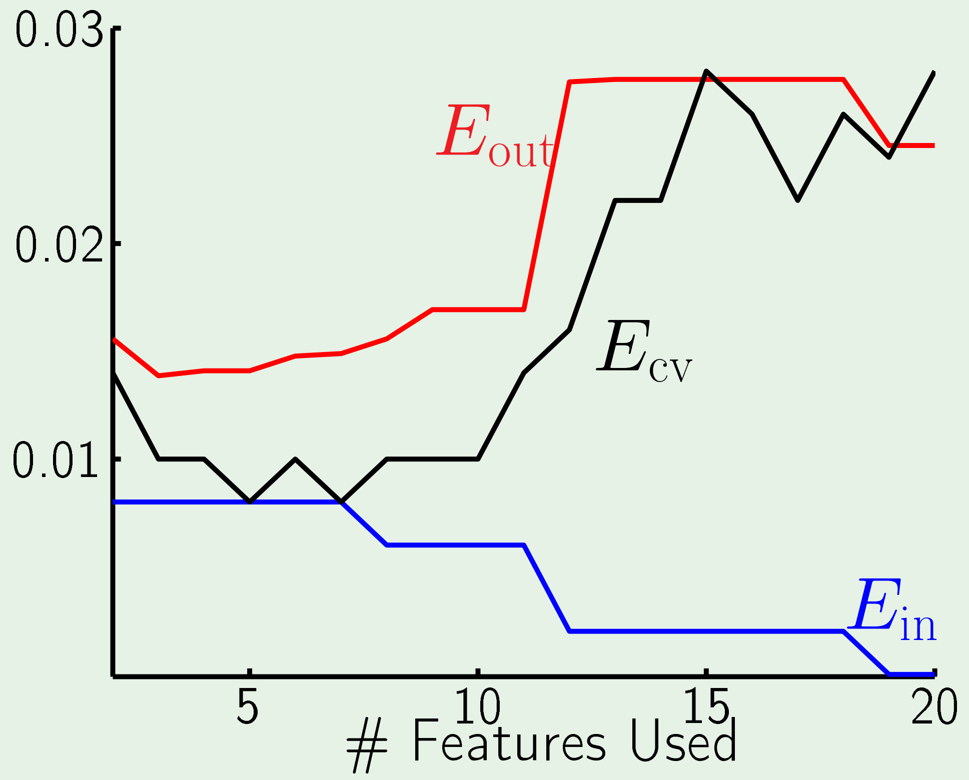
Cross validation in action

Digits classification task



Different errors

Eout from test set, not used to choose anything



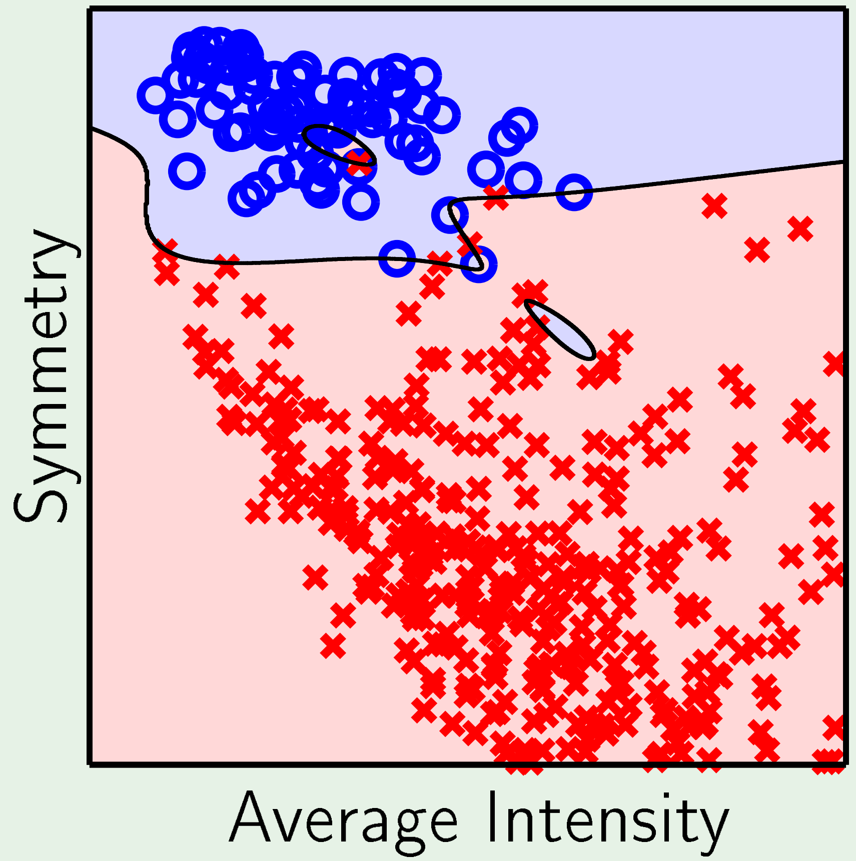
$$(1, x_1, x_2) \rightarrow (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, \ldots, x_1^5, x_1^4x_2, x_1^3x_2^2, x_1^2x_2^3, x_1x_2^4, x_2^5)$$

Here, we have 500 training examples and use validation to choose were to cut off the transformation: we compare 20 models (1,x1), (1,x1,x2), etc. using cross-val (leave one out) to choose where to stop

The result

without validation

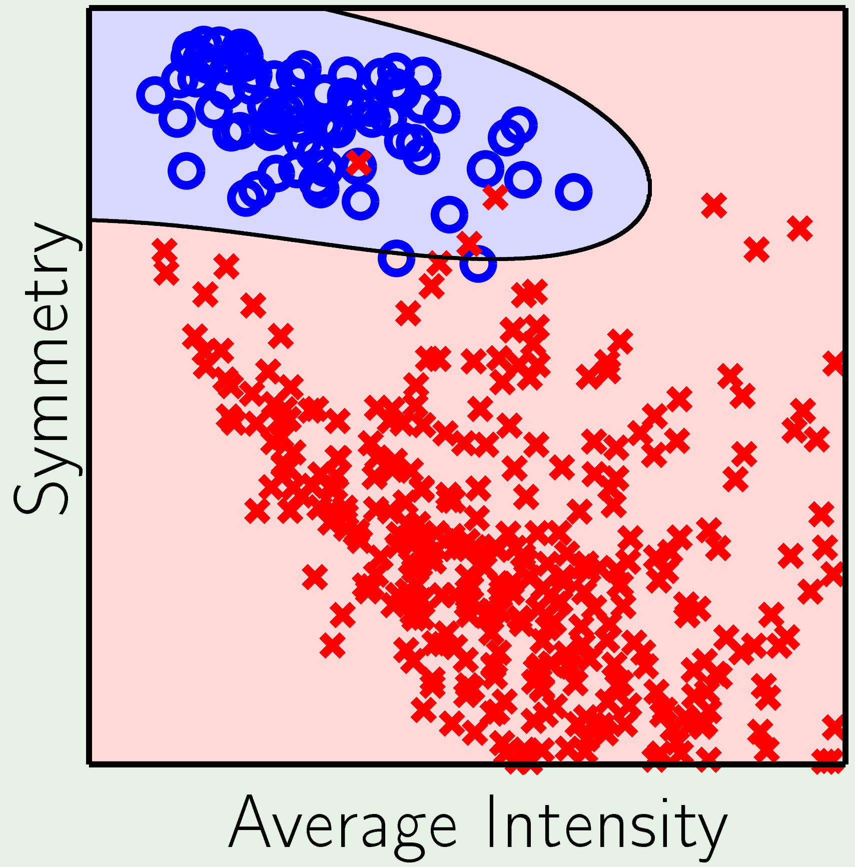
Using all 20 parameters to fit, so lots of overfitting and effort to handle all points perfectly, even anomalies which may not appear out of sample - like unregularized learning



$E_{\text{in}} = 0\%$ $\quad E_{\text{out}} = 2.5\%$

with validation

Using 6 parameters (minima of Ecv on error graph) we have a smooth surface, non-perfect error (but did not put effort where it did not belong)



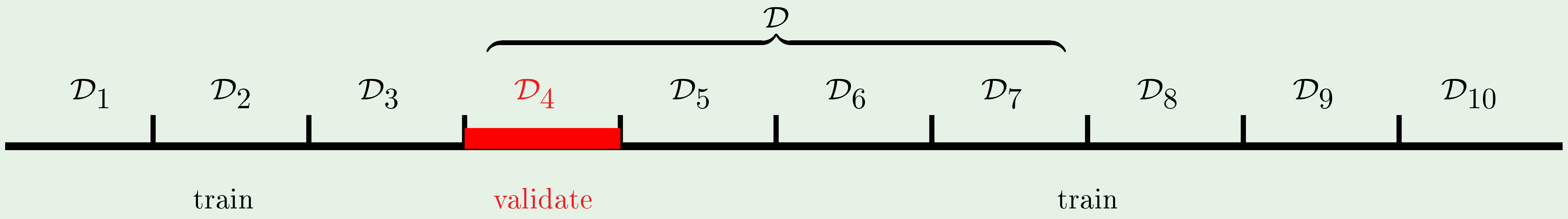
$E_{\text{in}} = 0.8\%$ $\quad E_{\text{out}} = 1.5\%$

40% improvement on Eout is a lot, given there is a limit we cannot exceed due to outliers etc. We can now see why validation in this context is considered similar to regularization: it does the same thing by preventing overfitting, but did so by estimating out-of-sample error rather than estimating something else. In most practical applications, we use both regularization and validation - very seldom can we get away with not using regularization or not using validation.

Leave more than one out

Leave one out: N training sessions on $N-1$ points each

More points for validation? We can take more points as long as N-K is still close to N so g- is close to g



$\frac{N}{K}$ training sessions on $N-K$ points each

Note that leave one out is essentially N-fold cross validation

10-fold cross validation: $K = \frac{N}{10}$

(10-fold works very well in practice)