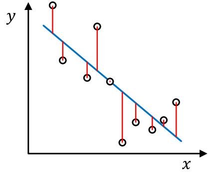
Review of lecture 2

- **Linear regression model**
  Predict real valued output $y \in \mathbb{R}$



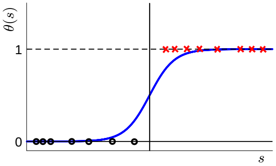
$$h(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x}$$

- **Linear regression algorithm**
  Minimize in-sample squared error

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)^{\mathsf{T}}\left(\mathbf{X}\mathbf{w} - \mathbf{y}\right)$$

$$\mathbf{w} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$$

- **Logistic regression model**
  Predict the probability of the class
  $y = +1$ given $\mathbf{x}$, $P(y = +1|\mathbf{x})$



$$h(\mathbf{x}) = \theta(\mathbf{w}^{\mathsf{T}}\mathbf{x})$$

$$\theta(\mathbf{w}^{\mathsf{T}}\mathbf{x}) = \frac{e^{\mathbf{w}^{\mathsf{T}}\mathbf{x}}}{1 + e^{\mathbf{w}^{\mathsf{T}}\mathbf{x}}}$$

- **Logistic regression algorithm**
  Minimize in-sample cross-entropy error

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + e^{-y_n \mathbf{w}^{\mathsf{T}}\mathbf{x}_n}\right)$$

- **Gradient descent**

Machine Learning: Lecture 3

Regularization
Validation

Mirko Mazzoleni

23 May 2017

University of Bergamo
Department of Management, Information and Production Engineering
mirko.mazzoleni@unibg.it

Outline

- Overfitting
- Regularization
- Validation
- Model selection
- Cross-validation

Outline

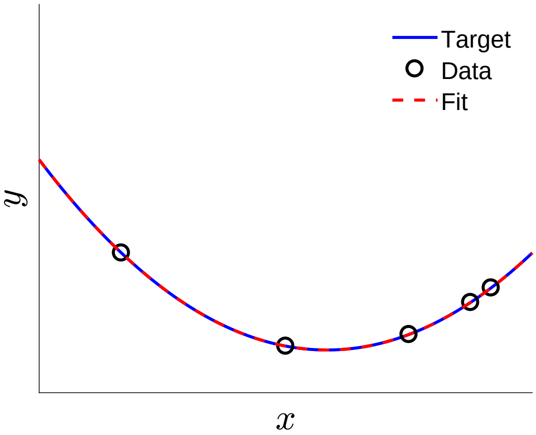
- **Overfitting**
- Regularization
- Validation
- Model selection
- Cross-validation

Overfitting

Overfitting

- Simple target function
- $N = 5$ points
- Fit with 4th order polynomial

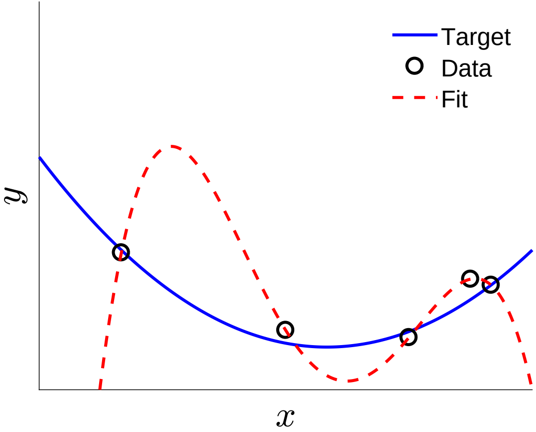
$$E_{\text{in}} = 0, \;\; E_{\text{out}} = 0$$



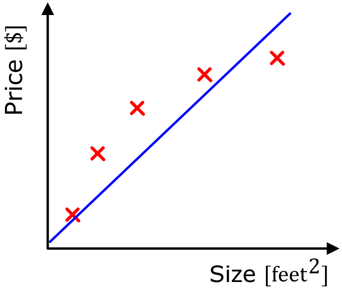
Overfitting

- Simple target function
- $N = 5$ **noisy** points
- Fit with 4th order polynomial

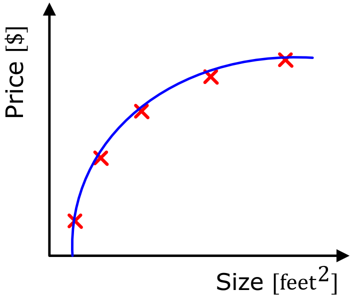
$E_{\text{in}} = 0, \;\; E_{\text{out}}$ is huge



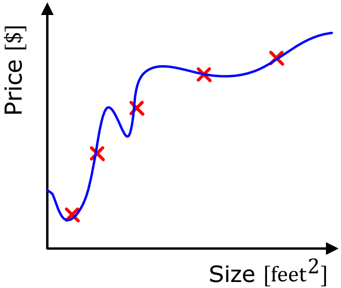
Overfitting



Underfit



Just right

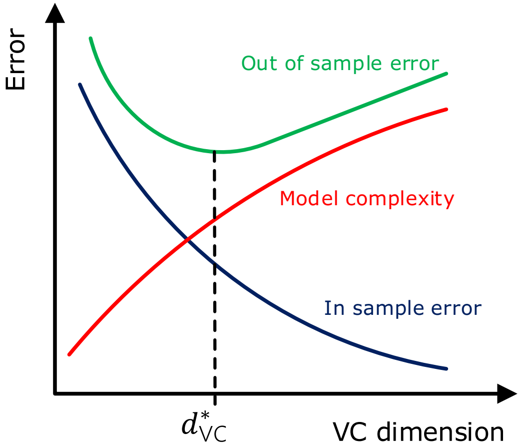


Overfit

Overfitting

We talk of overfitting when decreasing E_{in} leads to increasing E_{out}

- Major source of failure for machine learning systems
- Overfitting leads to bad generalization
- A model can exibit bad generalization even if it does not overfit



What causes overfitting

Overfitting occurs because the learning model spends its resources trying to fit the noise, on the finite dataset of size N

- The learning model is not able to discern the **signal** from the **noise**
- The effect of the noise is to show to the learning model "hallucinating patterns" that do not exist

Actually, two noise terms:

1. **Stochastic noise** $\rightarrow$ random noise on the observations
2. **Deterministic noise** $\rightarrow$ fixed quantity that depends on $\mathcal{H}$

Deterministic noise

The part of f that $\mathcal{H}$ cannot capture: $f(\mathbf{x}) - h^*(\mathbf{x})$

- $h^*(\mathbf{x})$ is the best hypothesis in $\mathcal{H}$ for approximating f
- Because h^* is simpler than f, any part of f that h^* cannot explain is like a noise for h^*, since that points deviate from h^* for an unknown reason

Overfitting can happen even if the target is noiseless (no stochastic noise)

- On a finite dataset N, the deterministic noise can led the learning model to interpret the data in a wrong way
- This happens because the model is not able to "understand" the target, and so it "misinterprets" the data, constructing its own hypothesis, which is wrong because it did not understand the true function given its limited hypothesis space

Overfitting behaviour

Summarising we have that:

- Overfitting $\uparrow$ if stochastic noise $\uparrow$ $\rightarrow$ the model is deceived by erroneous data
- Overfitting $\uparrow$ if deterministic noise $\uparrow$ $\rightarrow$ deterministic noise $\uparrow$ if target complexity $\uparrow$
- Overfitting $\downarrow$ if N $\uparrow$ $\rightarrow$ with more data, it is more difficult that the model will follow the noise of all of them

Overfitting behaviour with deterministic noise

Case study 1

Suppose $\mathcal{H}$ fixed, and increase the complexity of f

- Deterministic noise $\uparrow$, since there are more parts of f that h^* cannot explain, and then they act as noise for h^*
- Deterministic noise $\uparrow \implies$ overfitting $\uparrow$

Case study 2

Suppose f fixed, and decrease the complexity of $\mathcal{H}$

- Deterministic noise $\uparrow \implies$ overfitting $\uparrow$
- Simpler model $\implies$ overfitting $\downarrow$
- Most of the times the gain in reducing the model complexity exceeds the increase in deterministic noise

Bias - variance tradeoff revisited

Let the stochastic noise $\varepsilon(\mathbf{x})$ be a random variable with mean zero and variance σ^2

$$\mathbb{E}_{\mathcal{D},\mathbf{x},\varepsilon}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - (f(\mathbf{x}) + \varepsilon(\mathbf{x}))\right)^2\right] =$$

$$\underbrace{\mathbb{E}_{\mathcal{D},\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]}_{\textbf{var}} + \underbrace{\mathbb{E}_{\mathbf{x}}\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2}_{\textbf{bias}} + \underbrace{\mathbb{E}_{\varepsilon,\mathbf{x}}\left[\left(\varepsilon(\mathbf{x})\right)^2\right]}_{\boldsymbol{\sigma^2}}$$

- Stochastic noise is related to power of the random noise $\rightarrow$ irreducible error
- Deterministic noise is related to **bias** $\rightarrow$ part of f that $\mathcal{H}$ cannot capture
- Overfitting is caused by **variance**. Variance is, in turn, affected by the noise terms, capturing a model's susceptibility to being led astray by the noise

Outline

- Overfitting
- **Regularization**
- Validation
- Model selection
- Cross-validation

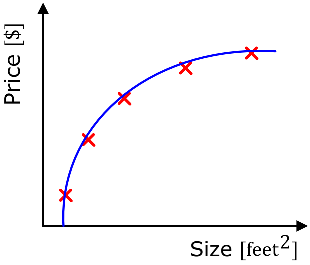
A cure for overfitting

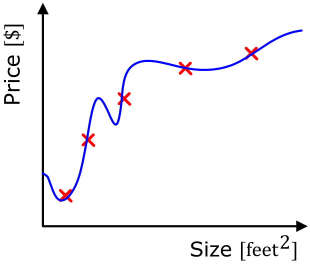
Regularization is the first line of defense against overfitting

- We have seen that complex model are more prone to overfitting
- This is because they are more powerful, and thus they can fit the noise
- Simple models exibits less variance because of their limited expressivity. This gain in variance often is greater than their greater bias
- However, if we stick only to simple models, we may not end up with a satisfying approximation of the target function f

How can we retain the benefits of both worlds?

A cure for overfitting





$$\mathcal{H}_2 : w_0 + w_1 x_1 + w_2 x^2 \qquad \mathcal{H}_4 : w_0 + w_1 x_1 + w_2 x^2 + w_3 x^3 + w_4 x^4$$

- We can recover the model $\mathcal{H}_2$ from the model $\mathcal{H}_4$ by imposing $w_3 = w_4 = 0$

$$\arg\min_{\mathbf{w}} \; \frac{1}{N} \sum_{n=1}^{N} \Big(h(\mathbf{x}_n; \mathbf{w}) - f(\mathbf{x}_n) \Big)^2 + 1000 \cdot (w_3)^2 + 1000 \cdot (w_4)^2$$

A cure for overfitting

$$\arg\min_{\mathbf{w}} \ \frac{1}{N}\sum_{n=1}^{N}\Big(h(\mathbf{x}_n;\mathbf{w}) - f(\mathbf{x}_n)\Big)^2 + \underbrace{1000\cdot(w_3)^2 + 1000\cdot(w_4)^2}_{\Omega}$$

- The cost function has been augmented with a penalization term $\Omega(w_3, w_4)$
- The minimization algorithm now has to minimize both E_{in} and $\Omega(w_3, w_4)$
- Due to the minimization process the value of w_3 and w_4 will be shrinked toward a small value $\rightarrow$ not exactly zero: soft order constraint
- If this value is very small, then the contribution of x_3 and x_4 is negligible $\rightarrow$ w_3 and w_4 (very small) multiply x_3 and x_4
- In this way, we ended up with a model that it is like the model $\mathcal{H}_2$ in terms of complexity $\rightarrow$ we can think as like the features x_3 and x_4 were not present
- It is like we reduced the number of parameters of the $\mathcal{H}_4$ model

Regularization

The concept introduced in the previous slides can be extented on the entire model's parameters

Instead of minimizing the in-sample error E_{in}, minimize the **augmented error**:

$$\boxed{E_{\text{aug}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\Big(h(\mathbf{x}_n;\mathbf{w}) - f(\mathbf{x}_n)\Big)^2 + \lambda\sum_{j=1}^{d}(w_j)^2}$$

- Usually we do not want to penalize the intercept w_0, so j starts from 1
- The term $\Omega(h) = \sum_{j=1}^{d}(w_j)^2$ is called regularizer
- The regularizer is a penalty term which depends on the hypothesis h
- The term λ weights the importance of minimizing E_{in}, with respect to minimizing $\Omega(h)$

Regularization

The minimization of E_{aug} can be viewed as a constrained minimization problem

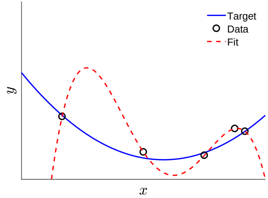
$$\text{Minimize } \; E_{\text{aug}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\Big(h(\mathbf{x}_n;\mathbf{w}) - f(\mathbf{x}_n)\Big)^2$$

$$\text{Subject to : } \; \mathbf{w}^{\mathsf{T}}\mathbf{w} \leq C$$

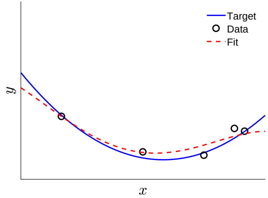
- With this view, we are explicitly constraining the weights to not have certain large values
- There is a relation between C and λ in such a way that if $C \uparrow$ the $\lambda \downarrow$
- Infact, bigger C means that the weights can be greater. This is equal to set for a lower λ, because the regularization term will be less important, and therefore the weights will not be shrunked as much

Effect of λ

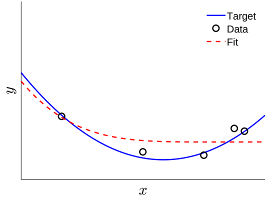
λ_1



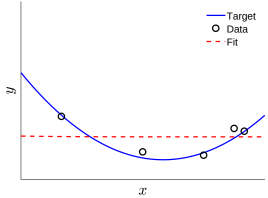
$\lambda_2 > \lambda_1$



$\lambda_3 > \lambda_2$



$\lambda_4 > \lambda_3$



Overfit $\rightarrow$ $\rightarrow$ **Underfit**

Augmented error

General form of the augmented error

$$E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda\Omega(h)$$

Recalling the VC-generalization bound

$$E_{\text{out}}(\mathbf{w}) \leq E_{\text{in}}(\mathbf{w}) + \Omega(\mathcal{H})$$

- $\Omega(h)$ is a measure of complexity of a specific hypothesis $h \in \mathcal{H}$
- $\Omega(\mathcal{H})$ measures the complexity of the hypothesis space $\mathcal{H}$
- The two quantities are obviously related, in the sense that a more complex hypothesis space $\mathcal{H}$ is described by more complex function h

The augmented error E_{aug} is **better** than E_{in} as a proxy for E_{out}

Augmented error

The holy Grail of machine learning would be to have a formula for E_{out} to minimize

- In this way, it would be possible to directly minimize the out of sample error instead of the in sample one
- Regularization helps by estimating the quantity $\Omega(h)$, which, added to E_{in}, gives E_{aug}, an estimation of E_{out}

Choice of the regularizer

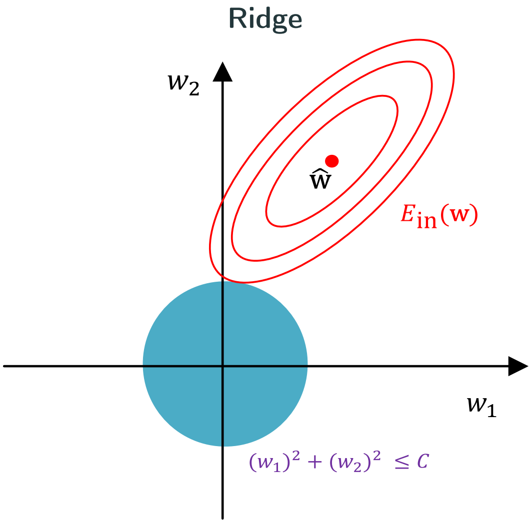
There are many choices of possible regularires. The most used ones are:

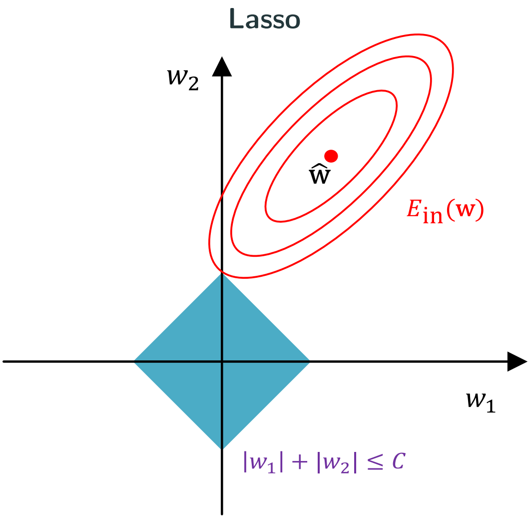
- $\boldsymbol{L_2}$ **regularizer:** also called Ridge regression, $\Omega(h) = \sum_{j=1}^{d} w_j^2$
- $\boldsymbol{L_1}$ **regularizer:** also called Lasso regression, $\Omega(h) = \sum_{j=1}^{d} |w_j|$
- **Elastic-net regularizer:** $\Omega(h) = \sum_{j=1}^{d} \alpha w_j^2 + (1-\alpha)|w_j|$

The different regularizers behaves differently:

- The ridge penalty tends to shrink all coefficients to a lower value
- The lasso penalty tends to set more coefficients exactly to zero
- The elastic-net penalty is a compromise betweem ridge and lasso, with the α value controlling the two contributions

Geometrical interpretation



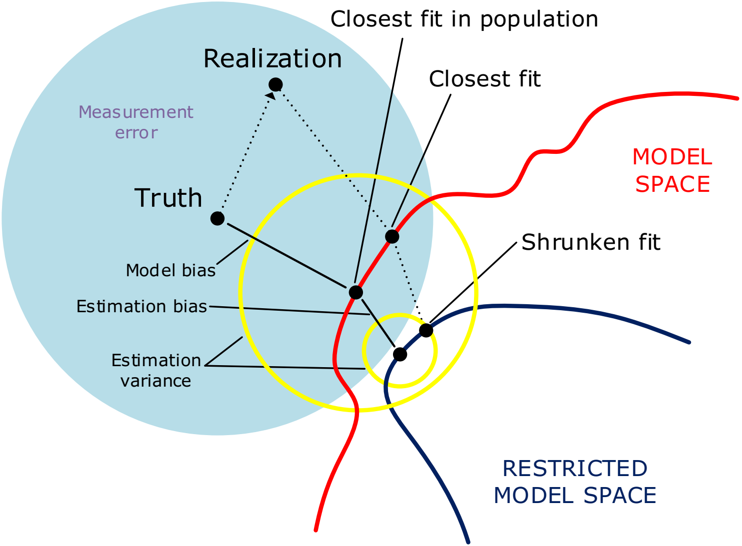


Regularization and bias-variance

The effects of the regularization procedure can be observed in the bias and variance terms

- Regularization trades more bias in order to considerely decrease the variance of the model
- Regularization strives for smoother hypothesis, thereby reducing the opportunities to overfit
- The amount of regularization λ has to be chosen specifically for each type of regularizer
- Usually λ is chosen by cross-validation
- When there is more stochastic noise or more deterministic noise, a higher value of λ is required to contrast their effect

Regularization and bias-variance: linear model case



Outline

- Overfitting
- Regularization
- **Validation**
- Model selection
- Cross-validation

Validation vs. 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

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

Validation set

The idea of a validation set is to estimate the model's performance out of sample

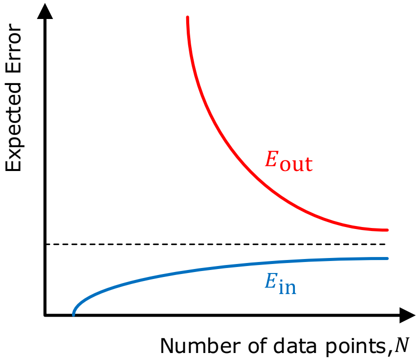
1. Remove a subset from the training data $\rightarrow$ this subset is not used in training
2. Train the model on the remaining training data $\rightarrow$ the model will be trained on *less* data
3. Evaluate the model's performance on the held-out set $\rightarrow$ this is an unbiased estimation of the out of sample error
4. Retrain the model on *all* the data

K is taken out of N

Given the dataset $\mathcal{D} = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

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

- Small K: bad estimate of E_{out}
- Large K: possibility of learning a bad model (learning curve)



K is put back into N

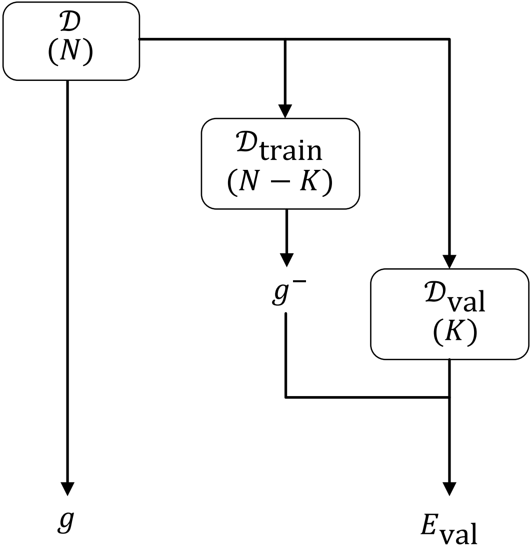
$$\begin{array}{cccc} \mathcal{D} & \rightarrow \quad \mathcal{D}_{\text{train}} & \cup & \mathcal{D}_{\text{val}} \\ \downarrow & \downarrow & & \downarrow \\ N & N-K & & K \end{array}$$

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

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

Rule of thumb:

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



Outline

- Overfitting
- Regularization
- Validation
- **Model selection**
- Cross-validation

Model selection

The most important use of a validation set is model selection

- Choose between a linear model and a nonlinear one
- Choice of the order of the polynomial in a model
- Choice of the regularization parameter
- Any other choice that affects the model learning

If the validation set is used to perform choices (e.g. to select the regularization parameter λ), then it **no longer** provides an unbiased estimate of E_{out}

There is the need of a third dataset: the **test set**, onto which to measure the model's performance E_{test}

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

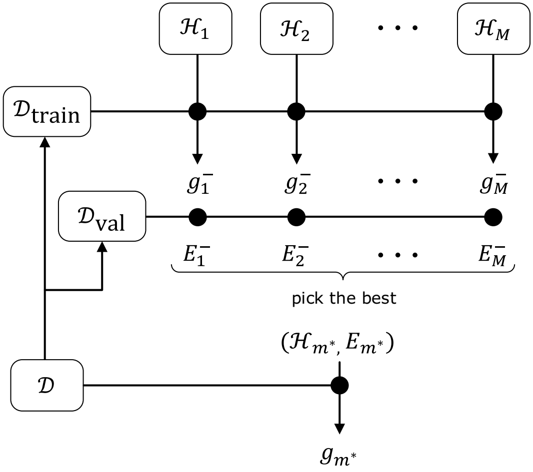
M models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$

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

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

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

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



How much bias

For the M models $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_M$, $\mathcal{D}_{\text{val}}$ is used for "training" on the **finalist model set**:

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

- The validation performance of the final model is $E_{\text{val}}\left(g_{m^*}^-\right)$
- This quantity is biased and not representative of $E_{\text{out}}\left(g_{m^*}^-\right)$, just as the in sample error E_{in} was not representative of E_{out} in the VC-analysis
- What happened is that $\mathcal{D}_{\text{val}}$ has become the "training set" for $\mathcal{H}_{\text{val}}$
- The risk is to overfit the validation set

In order to have a good match between $E_{\text{val}}\left(g_{m*}^-\right)$ and $E_{\text{out}}\left(g_{m*}^-\right)$, one has to have a number of validation data K sufficient for the number of parameters to set

- For 2 parameters, $K = 100$ is a good number

Data contamination

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

Contamination: Optimistic bias in estimating $E_{\rm out}$

- **Training set:** totally contaminated
- **Validation set:** slightly contaminated
- **Test set:** totally 'clean'

Outline

- Overfitting
- Regularization
- Validation
- Model selection
- **Cross-validation**

The dilemma about K

The following chain of reasoning:

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

highlights the dilemma in selecting K

Can we have K both small and large?

Leave one out cross-validation

Use $N-1$ points for training and $K=1$ point for validation

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

where $\mathcal{D}_n$ is the training set without the point n

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

The validation error on the unique point $\mathbf{x}_n$ is $e_n = E_{\text{val}}(g_n^-) = e(g_n^-(\mathbf{x}_n), y_n)$

It is then possible to define the **cross-validation error**

$$\boxed{E_{\text{cv}} = \frac{1}{N}\sum_{n=1}^{N} e_n}$$

Cross-validation for model selection

Cross-validation can be used effectively to perform model selection by selecting the right regularization parameter λ

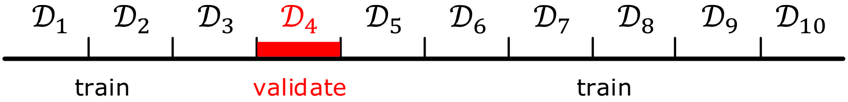
1. Define M models by choosing different values for λ: $(\mathcal{H}, \lambda_1), (\mathcal{H}, \lambda_2), \ldots, (\mathcal{H}, \lambda_M)$
2. **for** each model $m = 1, \ldots, M$ **do**
   2.1 Use cross-validation to obtain estimates of the out of sample error for each model
3. Select the model m^* with the smallest cross-validation error $E_{\text{cv}}(m^*)$
4. Use the model $(\mathcal{H}, \lambda_{m^*})$ and all the data $\mathcal{D}$ to obtain the final hypothesis g_{m^*}

Leave more than one out

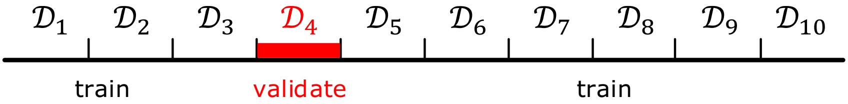
Leave-one-out cross-validation as the disadvantage that:

- It is computationally expensive, requiring a total of N training sessions for each of the M models
- The estimated cross-validation error has high variance, since it is based only on one point

It is possible to reserve more points for validation by dividing the training set in “folds”



Leave more than one out



- This produces $\frac{N}{K}$ training session on $N - K$ points each
- A good comprosime for the number of folds is 10

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

- Pay attention to not reduce the training set to much
  1. Look at the learning curves
  2. Look at the number of parameters (related to VC-dimension)