Lecture 5:
Model selection and validation

Machine Learning 2025

Federico Chiariotti (federico.chiariotti@unipd.it)

Università
degli Studi
di Padova

Lecture plan

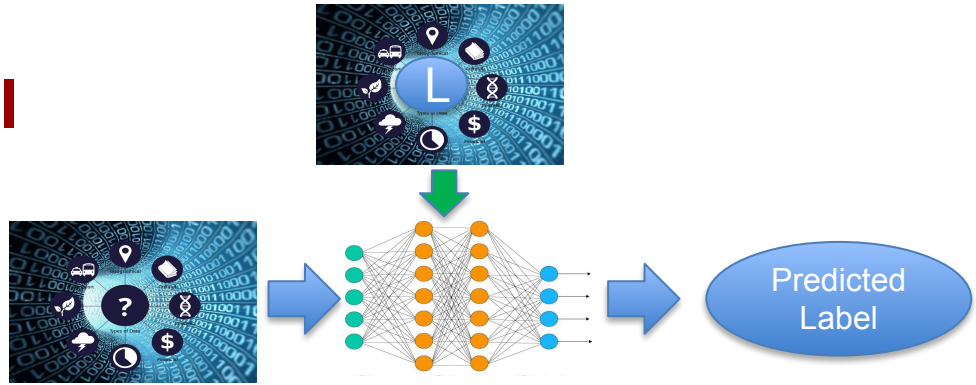
Date	#	Topic	Date	#	Topic	Date	#	Topic
Sep. 30	1	Introduction	Nov. 4	L2	*Model selection*	Nov. 28	12	CNNs
Oct. 7	2	PAC learning	Nov. 7	8	SVMs	???	13	PCA
Oct. 10	3	Uniform convergence	Nov. 11	L3	*Linear models*	Dec. 5	14	Clustering models
Oct. 14	L1	*Python basics*	???	9	Kernels	Dec. 9	L6	*Neural networks*
Oct. 17	4	VC dimension	Nov. 14	10	Ensemble models	Dec. 16	L7	*Clustering*
Oct. 21	5	Model selection	Nov. 18	L4	*SVMs*	Dec. 19	15	Reinforcement
Oct. 24	6	Linear classification	Nov. 21	11	Neural networks	???	L8	*Reinforcement*
Oct. 31	7	Linear regression	Nov. 25	L5	*Random forests*	???	16	Exercises and Q&A

IMPORTANT: no lecture on October 28!

Recap

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Supervised learning model



- Prediction rule $h : X \rightarrow Y$
  - The learner's output (hypothesis)
  - $A(S)$: hypothesis produced by algorithm A when it is fed training set S
- Data generation model
  - D is a distribution over X **(unknown to the machine learning algorithm)**
  - Instances are labeled according to $f : X \rightarrow Y$ **(unknown to the ML algorithm)**
  - Training set: sampling according to $D, y_i = f(x_i)\ \forall x_i \in S$
- Success metric
  - Classifier error: probability of predicting the wrong label over D

Empirical Risk Minimization (ERM)

- ➔ Learner outputs $h_S : X \to Y$
- ➔ **Goal: find $\mathbf{h}^*$ that minimizes $\mathbf{L_{D,f}(h)}$**
- ➔ Both D and f are unknown!

ERM: we minimize the loss on the training set $L_S(h) = \frac{\sum_{i=1}^{m} |h(x_i) - y_i|}{m}$

This works if we use a 0-1 loss: the definition can be generalized

The empirical risk is also called training error or training loss

Agnostic PAC learnability

Since we dropped the realizability assumption, we cannot reach 0 loss. What we can try and guarantee is a bound on the loss with respect to the minimum possible loss in the hypothesis class

A hypothesis class H is **agnostic** PAC learnable if there is a function $m_H : (0,1)^2 \to \mathbb{N}$ and a learning algorithm such that for every $\delta, \varepsilon \in (0,1)$ and for every learning problem D, the algorithm will satisfy the following condition over an IID training set of size $m \geq m_H$:

$$L_D(h) \leq \min_{h^* \in H} L_D(h^*) + \varepsilon$$

Uniform convergence and PAC learnability

A hypothesis class H has the uniform convergence property with respect to a domain Z and a loss function ℓ if there exists a function $m_H^{\mathrm{UC}} : (0,1)^2 \to \mathbb{N}$ such that for every $\varepsilon, \delta \in (0,1)$ and every distribution D, any set S of size $m \geq m_H^{\mathrm{UC}}(\varepsilon, \delta)$ is ε-representative with probability higher than $1 - \delta$

If a class H has the uniform convergence property with function m_H^{UC}, we have:

1. The class is agnostically PAC learnable with sample complexity

$$m_H(2\varepsilon, \delta) \leq m_H^{\mathrm{UC}}(\varepsilon, \delta)$$

2. ERM is a valid PAC learning algorithm for H

No free lunch theorem

Let A be any learning algorithm for binary classification, with 0-1 loss, over a domain X. Let m be any number smaller than $\frac{|X|}{2}$. There exists a distribution D over $X \times \{0,1\}$ such that:

1. There exists a function $f : X \to \{0,1\}$ with $L_D(f) = 0$
2. We have $L_D(A(S)) \geq \frac{1}{8}$ with probability $\frac{1}{7}$ over the choice of $S \sim D^m$

Corollary: if H is the set of all functions and X is an infinite set, we do **not** have PAC learnability

Definition: VC dimension

The Vapnik–Chervonenkis (VC) dimension of H is the maximum possible size d of a set $C = \{c_1, \ldots, c_d\} \subset X$ that can be shattered by H_C

In finite hypothesis classes, sets with a size larger than $\log_2(|H|)$ can never be shattered (there aren't enough functions to cover all points)

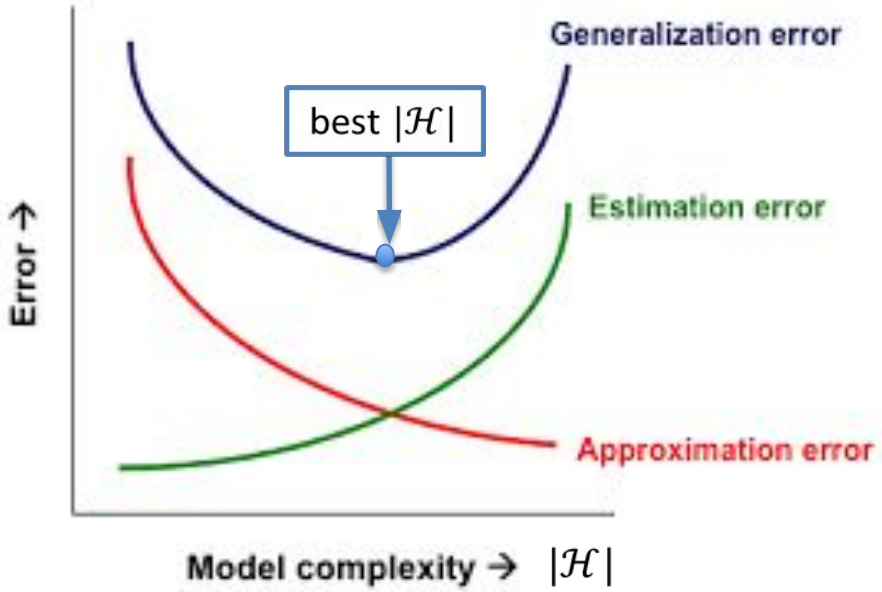
Infinite hypothesis classes may shatter sets of arbitrary size: if $VC(H) = \infty$, the class is not PAC learnable due to the corollary we just proved

The fundamental theorem of PAC learning

Let H be a hypothesis class for a binary classification problem with 0-1 loss. The following statements are equivalent:

1. H has the uniform convergence property
2. Any ERM rule is a successful agnostic PAC learner for H
3. H is agnostic PAC learnable
4. H is PAC learnable (under realizability condition)
5. Any ERM rule is a successful PAC learner for H (under realizability condition)
6. H has a finite VC dimension

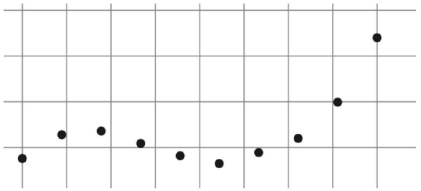
The bias-complexity trade-off

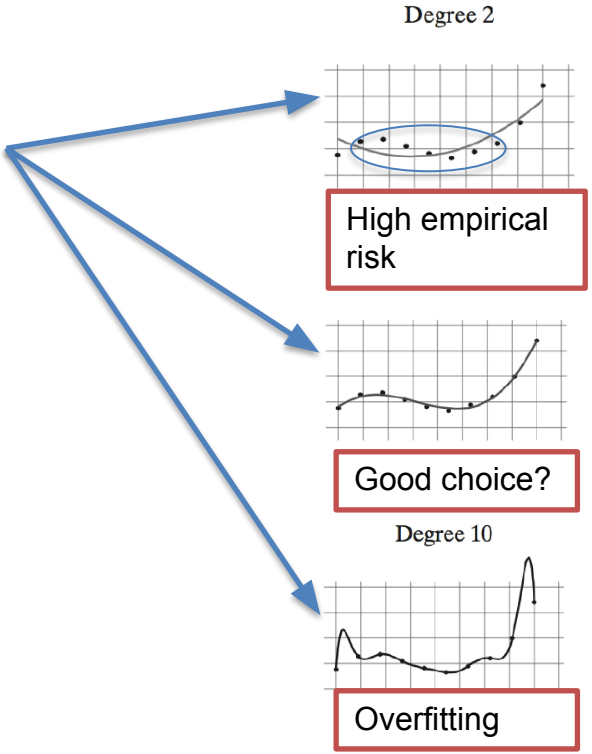


Part 1:
Structural Risk Minimization

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Choosing a good hypothesis class





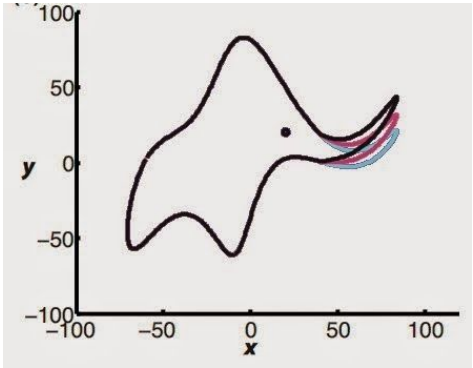
We can often use *hyperparameters* to select hypothesis classes: in polynomial regression, this is the degree of the function

How do we choose the correct degree?

Occam’s razor

All other things being equal, we should choose the simplest explanation (i.e., the smaller and simpler hypothesis class).

“With four parameters I can fit an elephant, and with five I can make him wiggle his trunk." (J. von Neumann)



Structural Risk

If $H = \cup_{n \in \mathbb{N}} H_n$, we can specify a *weighting function* $w : \mathbb{N} \to [0, 1]$ that reflects our preference for that subclass. If we have a finite number N of subclasses, choosing 1/N is equivalent to standard ERM. If every H_n has uniform convergence with function $m_{H_n}^{\mathrm{UC}}(\varepsilon, \delta)$, we define the minimum approximation error for a training set of size m:

$$\varepsilon_n(m, \delta) = \min \left\{ \varepsilon \in (0, 1) : m_{H_n}^{\mathrm{UC}}(\varepsilon, \delta) \leq m \right\}$$

Bounding Structural Risk

As for the previous results, we work with a bound to the structural risk, for which we can prove that the outcome is PAC learnable:

Let $w : \mathbb{N} \to [0, 1]$ be a function such that $\sum_{n=1}^{\infty} w(n) \leq 1$, and let $H = \cup_{n \in \mathbb{N}} H_n$, where every H_n satisfies the uniform convergence property. For a training set of size m and for any $\delta \in (0, 1)$ we have that, with probability at least $1 - \delta$:

$$L_D(h) \leq L_S(h) + 2 \min_{n:h \in H_n} \varepsilon_n(m, w(n)\delta) \ \forall h \in H$$

Proof: first step

By the definition of uniform convergence, for any m and $\delta \in (0,1)$ we have, with probability at least $1 - w(n)\delta$,

$$|L_D(h) - L_S(h)| \leq \varepsilon_n(m, w(n)\delta) \; \forall h \in H_n$$

The minimum requires this condition to hold for all values of n:

$$P\left[|L_D(h) - L_S(h)| > \min_{n:h\in H_n} |\varepsilon_n(m, w(n)\delta)\right] = P\left[\cup_{n:h\in H_n} |L_D(h) - L_S(h)| > \varepsilon_n(m, w(n)\delta)\right]$$

We can then apply the union bound:

$$P\left[\cup_{n:h\in H_n} |L_D(h) - L_S(h)| > \varepsilon_n(m, w(n)\delta)\right] \leq \sum_{n:h\in H_n} P\left[|L_D(h) - L_S(h)| > \varepsilon_n(m, w(n)\delta)\right]$$

Proof: second step

Bounded by $w(n)\delta$

We start from the union bound:

$$P\left[\cup_{n:h\in H_n}|L_D(h)-L_S(h)|>\varepsilon_n(m,w(n)\delta)\right]\leq \sum_{n:h\in H_n} P\left[|L_D(h)-L_S(h)|>\varepsilon_n(m,w(n)\delta)\right]$$

We then exploit the fact that $\sum_{n=1}^{\infty} w(n)\leq 1$ to obtain:

$$\sum_{n:h\in H_n} w(n)\delta \leq \sum_{n=1}^{\infty} w(n)\delta \leq \delta$$

By the relation between uniform convergence and PAC, we prove the bound

Structural Risk Minimization

The objective of SRM then becomes:

$$h_{\mathrm{SRM}} = \arg\min_{h \in H} L_S(h) + \varepsilon_{n(h)}(m, w(n_h)\delta)$$

where $n(h) = \min\{n : h \in H_n\}$. We added a term that accounts for our preference for some hypothesis subclasses to ERM, which allows us to apply Occam's razor.

Minimum Description Length

Let us put every hypothesis in a singleton class. By Hoeffding's inequality, this class has uniform convergence with $m^{\mathrm{UC}}(\varepsilon, \delta) = \frac{\log(2/\delta)}{\varepsilon^2}$. We then get

$$\varepsilon_h(m, \delta) = \sqrt{\frac{\log(2/\delta)}{2m}}$$

Suppose we have a way of describing a hypothesis using a prefix code. We use the length of the description as a weight:

$$h_{\mathrm{MDL}} = \arg\min_{h \in H} L_S(h) + \sqrt{\frac{|h| + \log(2/\delta)}{2m}}$$

Tikhonov Regularized Loss Minimization

If the hypothesis is defined by a parameter (or weight) vector $\mathbf{w}$, Tikhonov regularization uses the L2 norm of the weights $||\mathbf{w}||^2$:

$$\mathbf{w}_{\mathrm{RLM}} = \arg\min_{\mathbf{w}} L_S(\mathbf{w}) + \lambda||\mathbf{w}||^2$$

The parameter λ controls the trade-off between overfitting and underfitting:

➔ If λ is low, we are closer to ERM: the overfitting risk is higher, but empirical error is also low

➔ If λ is high, we have a strong preference for solutions with small weights, so we won't overfit, but the empirical error might be higher

Defining stability

We would like learning to be stable, i.e., to be unaffected by small changes in the training set: the smallest change is replacing one data point. We can then define an **On Average Replace One Stable (OAROS)** algorithm as:

Let $\varepsilon : \mathbb{N} \to \mathbb{R}$ be a monotonically decreasing function. A learning algorithm is OAROS with rate $\varepsilon(m)$ for distribution D if:

$$\mathbb{E}_{(S,z')\sim D^{m+1},i\sim U(m)}\left[\ell(A(S'),z_i) - \ell(A(S),z_i)\right] \leq \varepsilon(m)$$

OAROS

Let $\varepsilon : \mathbb{N} \to \mathbb{R}$ be a monotonically decreasing function. A learning algorithm is OAROS with rate $\varepsilon(m)$ for distribution D if:

$$\mathbb{E}_{(S,z')\sim D^{m+1}, i\sim U(m)}[\ell(A(S'), z_i) - \ell(A(S), z_i)] \leq \varepsilon(m)$$

Randomly choose sample to replace

Training set and extra sample from D

Replace z_i with z'

Stability theorem

If an algorithm is OAROS with rate $\varepsilon(m)$, we have:

$$\mathbb{E}_{S\sim D^m}\left[L_D(A(S)) - L_S(A(S))\right] \leq \varepsilon(m)$$

Bounding the difference between the empirical and true risk is a very useful property, which avoids overfitting and provides bounds for error (it's what we try to do with PAC!)

Proof

The empirical risk represents the expected value of the loss over a uniformly chosen point in the training set:

$$\mathbb{E}_S[L_S(A(S))] = \mathbb{E}_{S,i}[\ell(A(S), z_i)]$$

The true risk represents the expected value of the loss over a uniformly chosen point from the original distribution:

$$\mathbb{E}_S[L_D(A(S))] = \mathbb{E}_{S,z' \sim D}[\ell(A(S), z')] = \mathbb{E}_{S',z_i}[\ell(A(S'), z_i)]$$

If we combine the two, we get the theorem statement (expectation is linear)

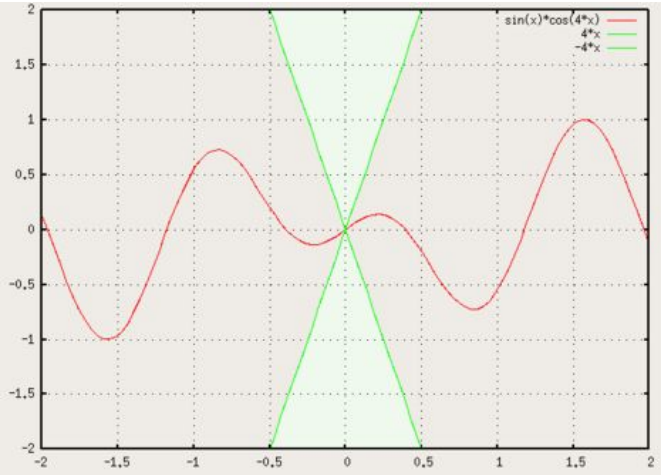
ρ-Lipschitz functions

A function $f : \mathbb{R}^d \to \mathbb{R}^k$ is ρ-Lipschitz over $C \subset \mathbb{R}^d$ if:

$$||f(\mathbf{w}_1) - f(\mathbf{w}_2)|| \leq \rho||\mathbf{w}_1 - \mathbf{w}_2|| \; \forall \mathbf{w}_1, \mathbf{w}_2 \in C$$

➔ The function cannot change too fast

➔ Linear bound on the derivative (if it exists)



Stability and Tikhonov regularization

If loss is convex, continuous, and ρ-Lipschitz, the RLM rule with Tikhonov regularization is OAROS with rate $\frac{2\rho^2}{\lambda m}$. Trivially,

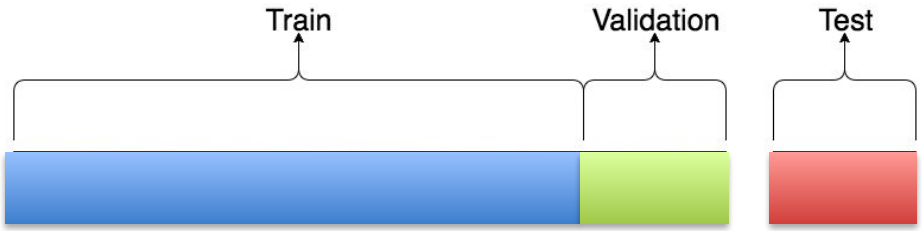
$$\mathbb{E}_{S\sim D^m}\left[L_D(A(S) - L_S(A(S))\right] \leq \frac{2\rho^2}{\lambda m}$$

We can control the bias-complexity trade-off using λ, which is a hyperparameter of the algorithm

Part 2:
Validation sets

Università
degli Studi
di Padova

Test and validation sets



We divide our data in 3 parts:

- ➔ The training set is used to choose the best hypothesis with ERM
- ➔ The validation set is used to choose the best hyperparameters or subclass
- ➔ The test set is used to estimate the true loss

Validation set approximation

If the validation set contains m_V samples, the validation loss (i.e., the empirical risk over the validation set) follows (with probability at least $1 - \delta$):

$$|L_D(h) - L_V(h)| \leq \sqrt{\frac{\log(2/\delta)}{2m_v}}$$

The validation set has not been used for training, so we do not have overfitting.

This is a better bound than the quantitative version of the fundamental theorem, which does not depend on the VC dimension

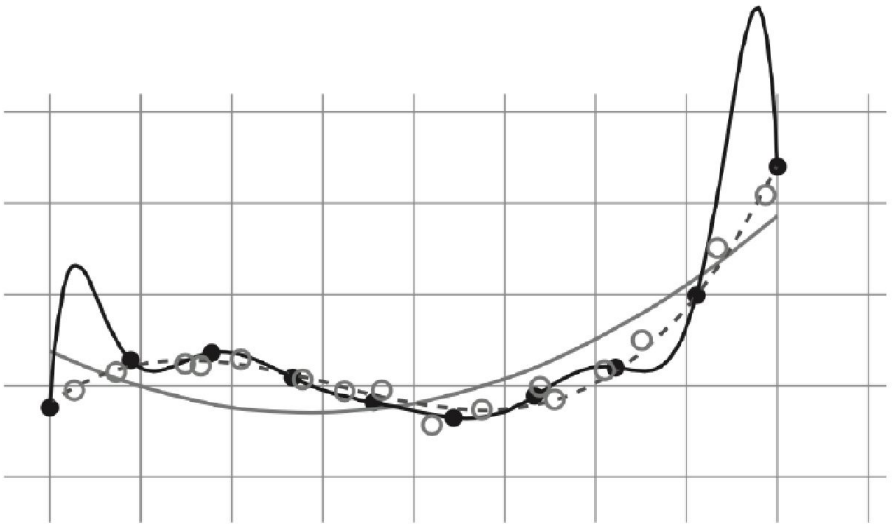
Model selection with validation

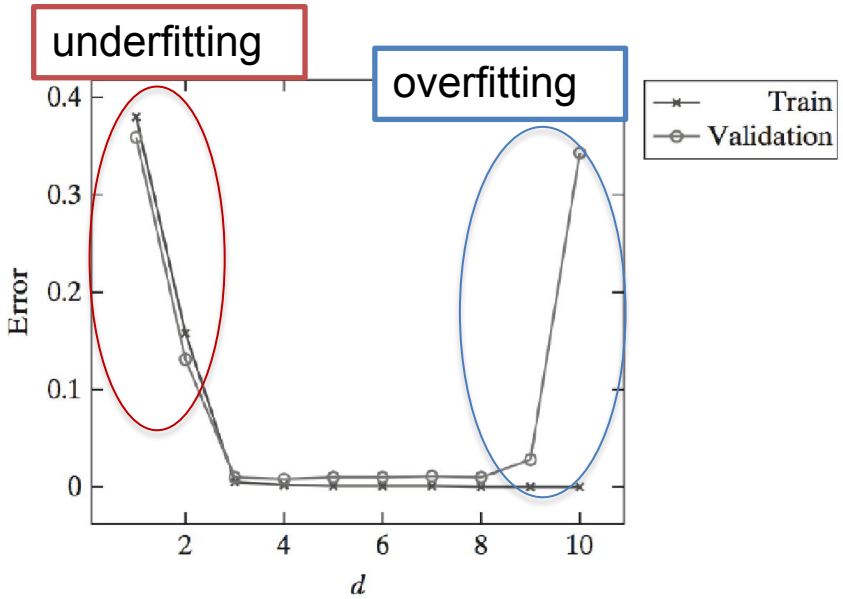
If we have a (finite) set of subclasses, i.e., $H = \cup_{n=1}^{N} H_n$, we can use the following algorithm:

1. Find the solution for each subclass (using only the **training set**),$h_{\mathrm{ERM},n}$
2. Collect the solutions in a new hypothesis class $H' = \cup_{n=1}^{N} \{h_{\mathrm{ERM},n}\}$
3. Select the final hypothesis by applying ERM over the **validation set**

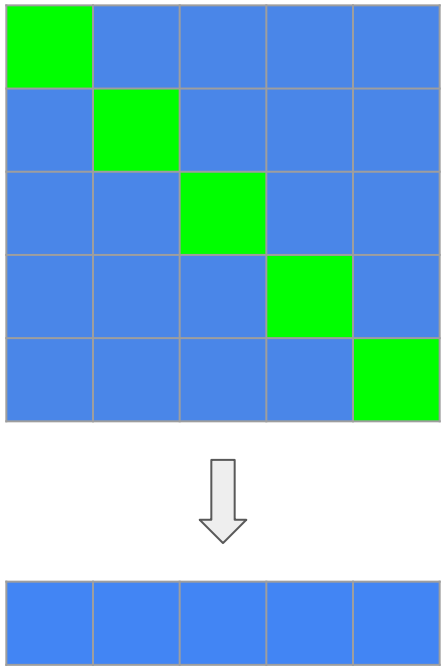
This may include different algorithms, or an algorithm with different hyperparameters

Model selection with validation





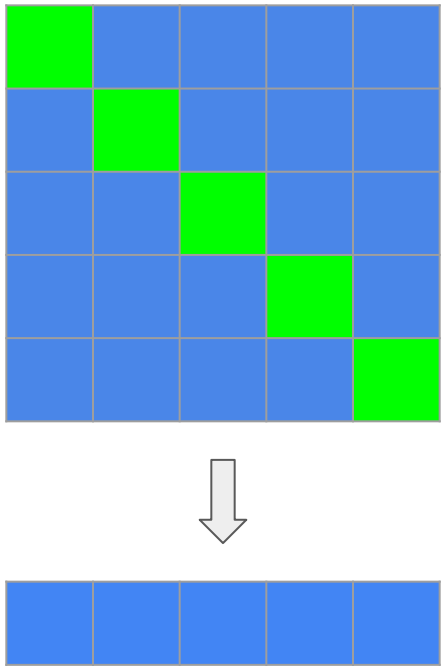
K-fold Cross Validation



Actually, we can reuse the validation set:

1. Split the training set in K *folds*
2. Perform validation using K-1 as the training set and the other one as the validation set
3. The **score** of each subclass is the average validation set
4. Choose the subclass with the highest score
5. Retrain on the whole training set

K-fold Cross Validation



k-Fold Cross Validation for Model Selection

input:
- training set $S = (\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
- set of parameter values Θ
- learning algorithm A
- integer k

partition S into $S_1, S_2, \ldots, S_k$

foreach $\theta \in \Theta$
- **for** $i = 1 \ldots k$
  - $h_{i,\theta} = A(S \setminus S_i; \theta)$
- $\text{error}(\theta) = \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\theta})$

output

$\theta^\star = \text{argmin}_\theta \left[\text{error}(\theta)\right]$

$h_{\theta^\star} = A(S; \theta^\star)$

Error decomposition

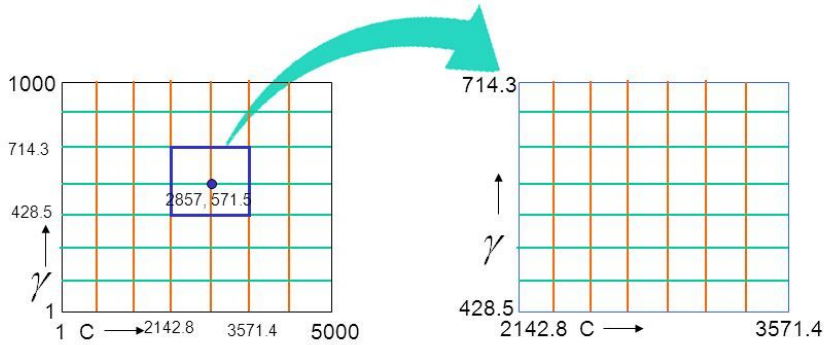
$$L_D(h_S) = (L_D(h_S) - L_V(h_S)) + (L_V(h_S) - L_S(h_S)) + L_S(h_S)$$

We can overfit in two ways:

1. Parameter overfitting: subclasses are too big for the training set
2. Hyperparameter overfitting: there are too many subclasses for the validation set

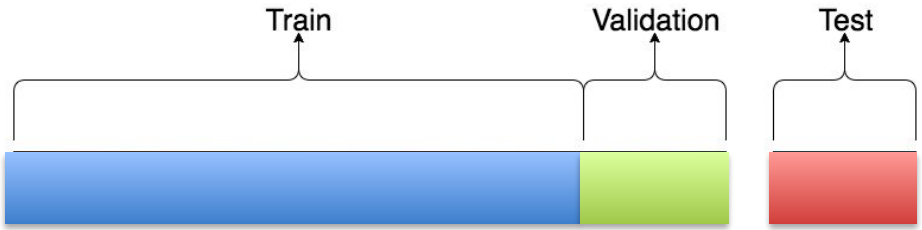
Keep in mind that the test loss is not the true loss, but rather the result of a sampling from a distribution: as such, it is normal to have some error if the test set is small (sometimes the test loss is even slightly smaller than the training loss!)

Grid search



- ➔ If we have multiple parameters, trying all combinations will lead to overfitting on the validation set (H' becomes huge, or even infinite!)
- ➔ We can start from a rough grid, then gradually increase precision
- ➔ Validation loss is **not** a good estimate of the true risk, as we are using it to choose a hypothesis class
- ➔ We need to use the test set to estimate the true risk

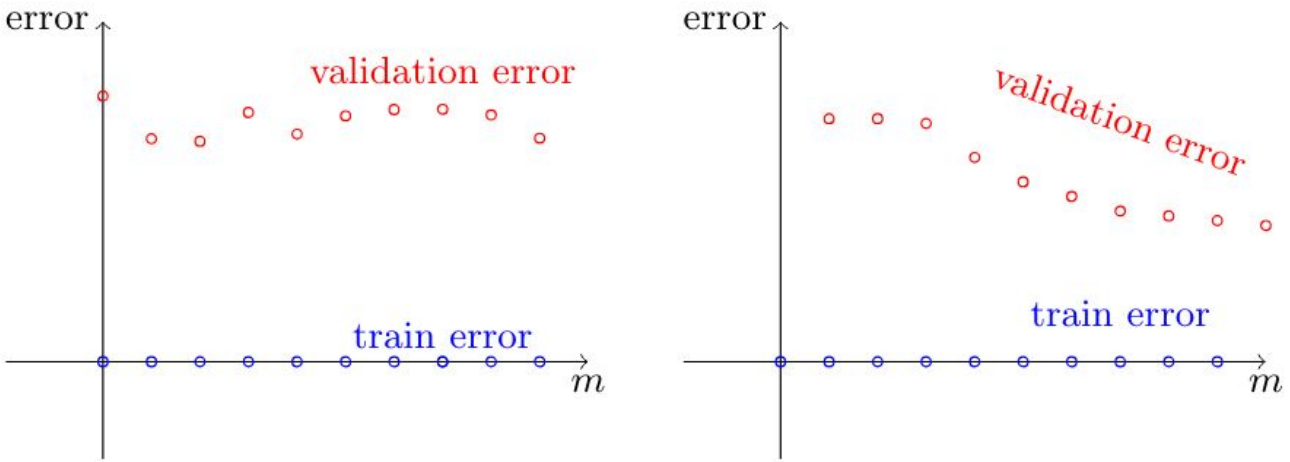
Test set contamination



The test set should be drawn from D, but it should be independent from the training (i.e., should not be used in any way, or contain the same samples)

If we use the test set to make decisions, we are overfitting on it: the only way to use it to estimate $L_D(h)$ without cheating is to **only use it at the end**. If the test set is contaminated (i.e., we use it during the learning process in any way), we need a new test set!

The training curve



First case: the number of samples is smaller than the VC dimension of the class! If we are overfitting (the training error is 0), we can at least get an idea whether we are learning something and will improve with more data, or whether we have to redesign our hypotheses and exploit prior domain knowledge