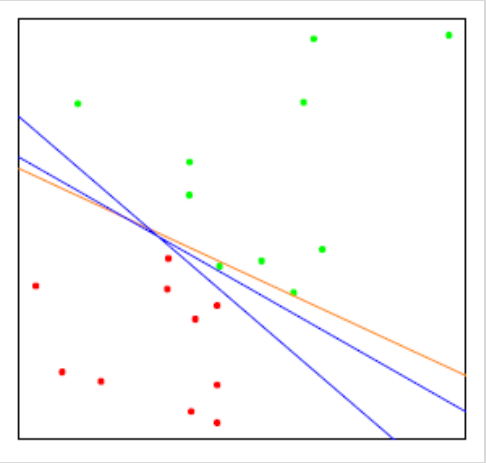


Parallel & Scalable Machine Learning

Introduction to Machine Learning Algorithms

Dr. – Ing. Morris Riedel
Adjunct Associated Professor
School of Engineering and Natural Sciences, University of Iceland
Research Group Leader, Juelich Supercomputing Centre, Germany

LECTURE 9

Validation and Regularization Techniques

March 8th, 2018
JSC, Germany

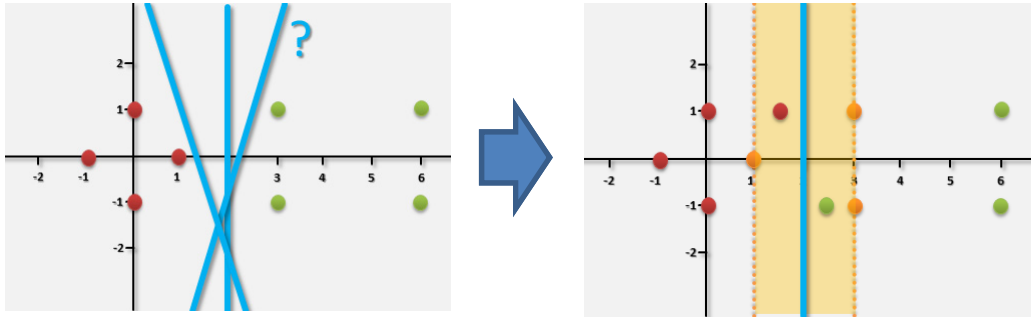
UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

JÜLICH Forschungszentrum
JÜLICH SUPERCOMPUTING CENTRE

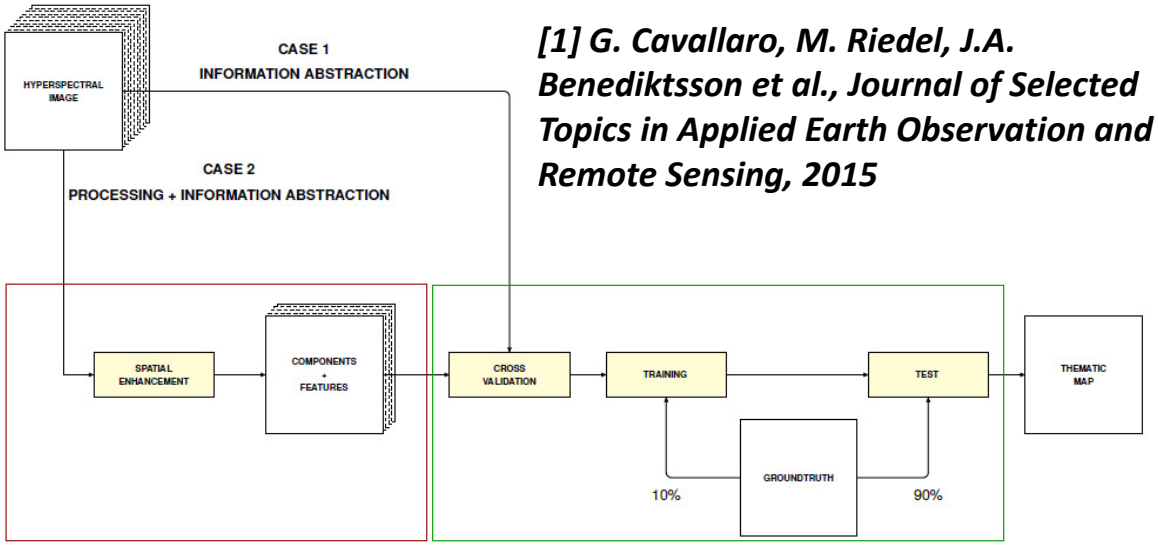
DEEP Projects

Review of Lecture 8

- Analysing Indian Pines Dataset
  - Remote Sensing Dataset
  - Raw & Processed
  - Using JURECA system with MPI-based piSVM 1.2.1 tool

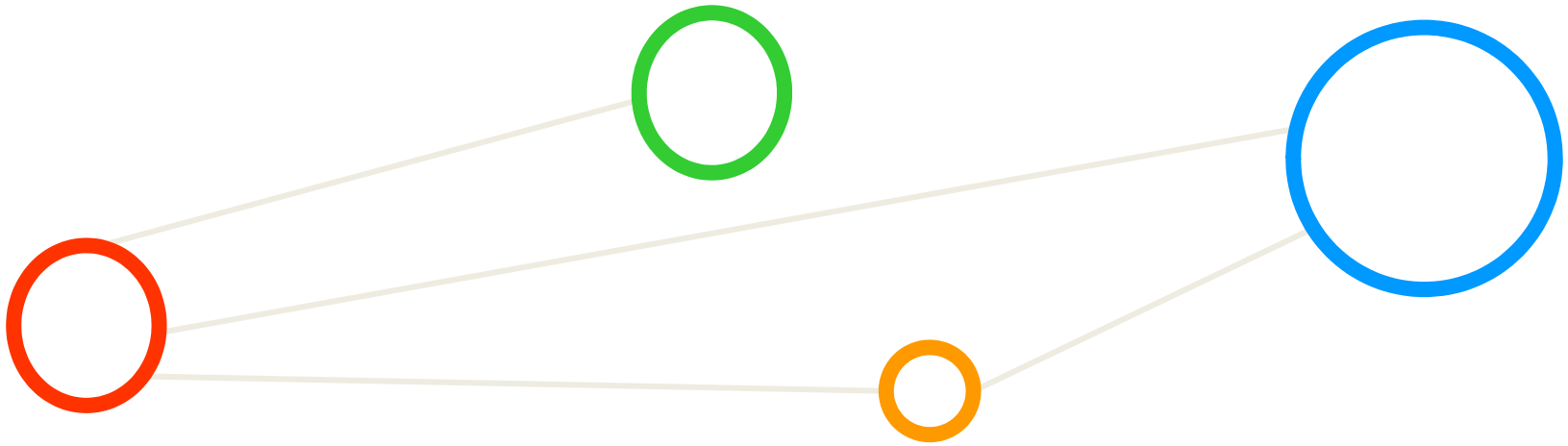


- Feature Engineering in Machine learning
  - Manual feature enhancements increased accuracy from ~44% to ~77%
  - Manual work before 'learning from data' is important



[1] G. Cavallaro, M. Riedel, J.A. Benediktsson et al., Journal of Selected Topics in Applied Earth Observation and Remote Sensing, 2015

Outline



Outline of the Course

1. Introduction to Machine Learning Fundamentals
2. PRACE and Parallel Computing Basics
3. Unsupervised Clustering and Applications
4. Unsupervised Clustering Challenges & Solutions
5. Supervised Classification and Learning Theory Basics
6. Classification Applications, Challenges, and Solutions
7. Support Vector Machines and Kernel Methods
8. Practicals with SVMs
9. Validation and Regularization Techniques
10. Practicals with Validation and Regularization
11. Parallelization Benefits
12. Cross-Validation Practicals

Day One – beginner

Day Two – moderate

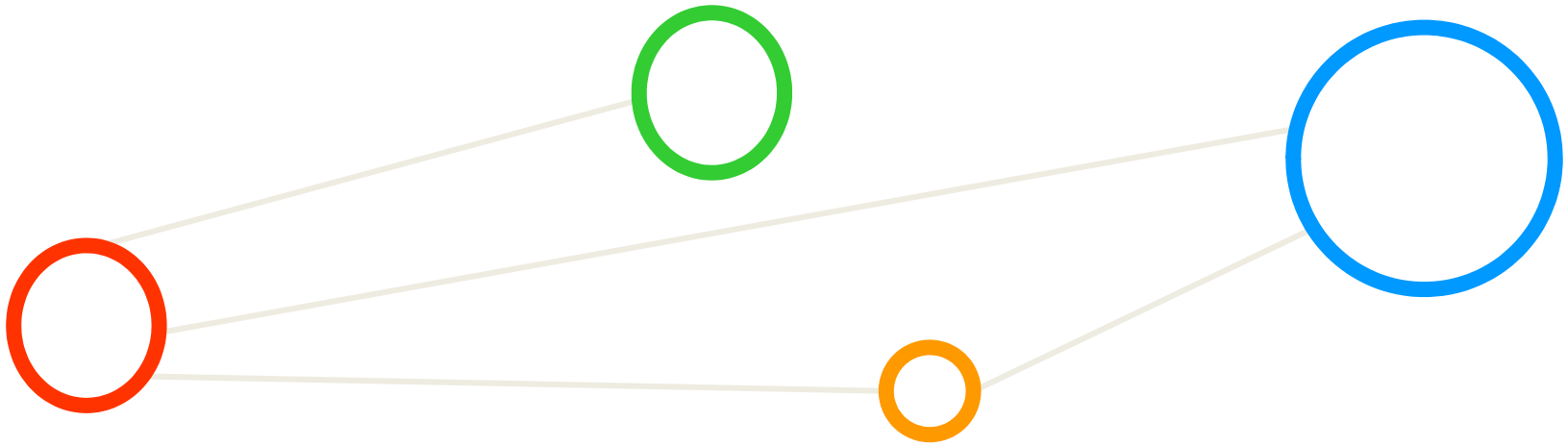
Day Three – expert

Outline

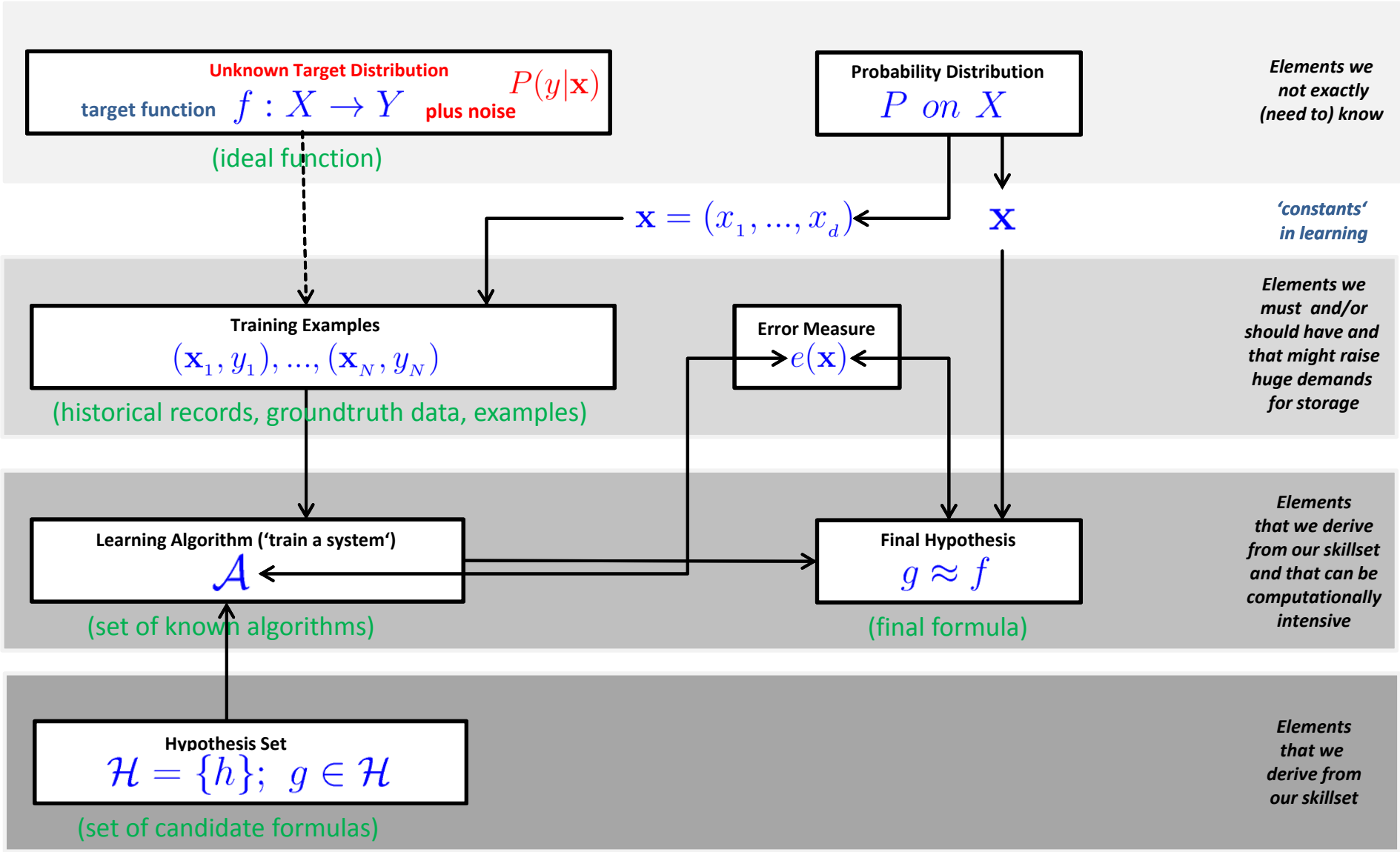
- Validation
  - Challenges to Pick the Right Model
  - Different Noise Elements in Overfitting
  - Validation for Model Selection
  - N-Fold Cross-Validation Technique
  - Validation of SVMs to Dataset Rome

- Regularization
  - Problem of Overfitting Revisited
  - Memorizing vs. Learning Models
  - Early Stopping in Training
  - Impacts of Regularization in SVMs
  - Regularization in Other Models

Validation



Mathematical Building Blocks – Revisited



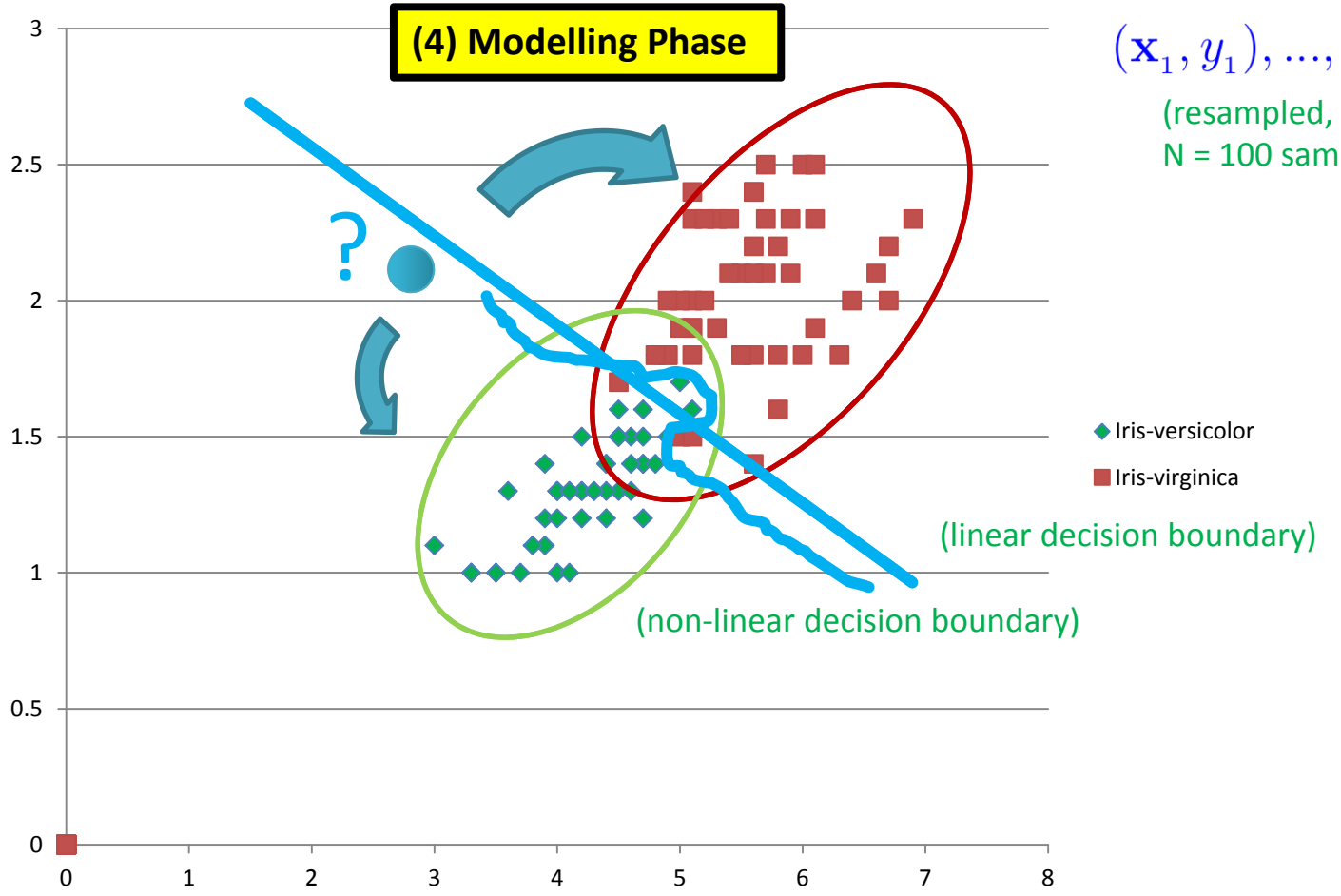
Key Challenges: Why is it not so easy in practice?

- Scalability
  - Gigabytes, Terabytes, and Petabytes datasets that fit not into memory
  - E.g. algorithms become necessary with out-of-core/CPU strategies
- High Dimensionality
  - Datasets with hundreds or thousand attributes become available
  - E.g. bioinformatics with gene expression data with thousand of features
- Heterogenous and Complex Data
  - More complex data objects emerge and unstructured data sets
  - E.g. Earth observation time-series data across the globe
- Data Ownership and Distribution
  - Distributed datasets are common (e.g. security and transfer challenges)

- **Key challenges faced when doing traditional data analysis and machine learning are scalability, high dimensionality of datasets, heterogenous and complex data, data ownership & distribution**

[2] Introduction to Data Mining

Non-linearly Seperable Data in Practice – Which model?



$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

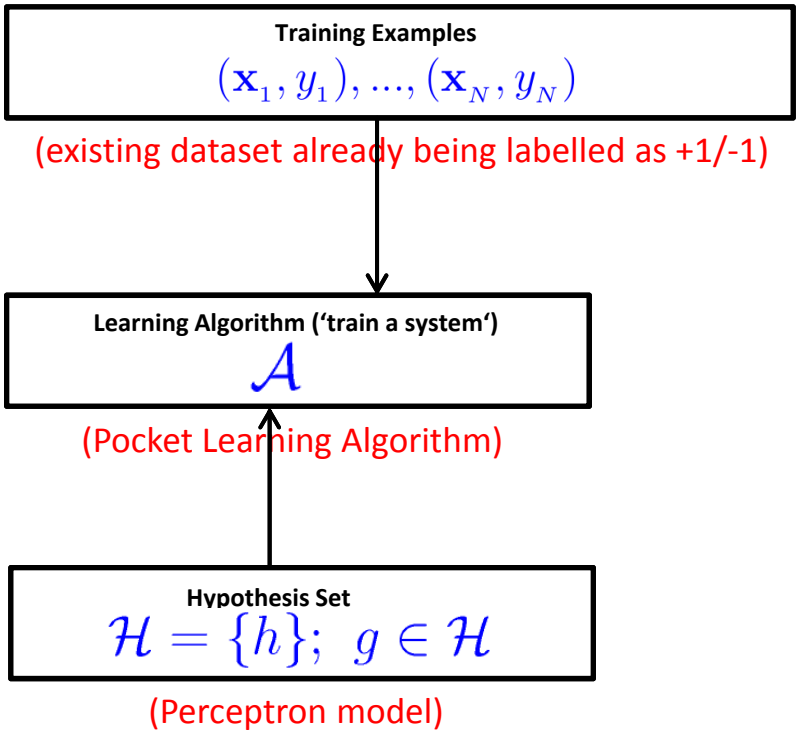
(resampled, again
N = 100 samples)

(lessons learned from practice: requires soft-thresholds to allow for some errors being overall better for new data
→ Occams razor – 'simple model better')

(lessons learned from practice: requires non-linear decision boundaries)

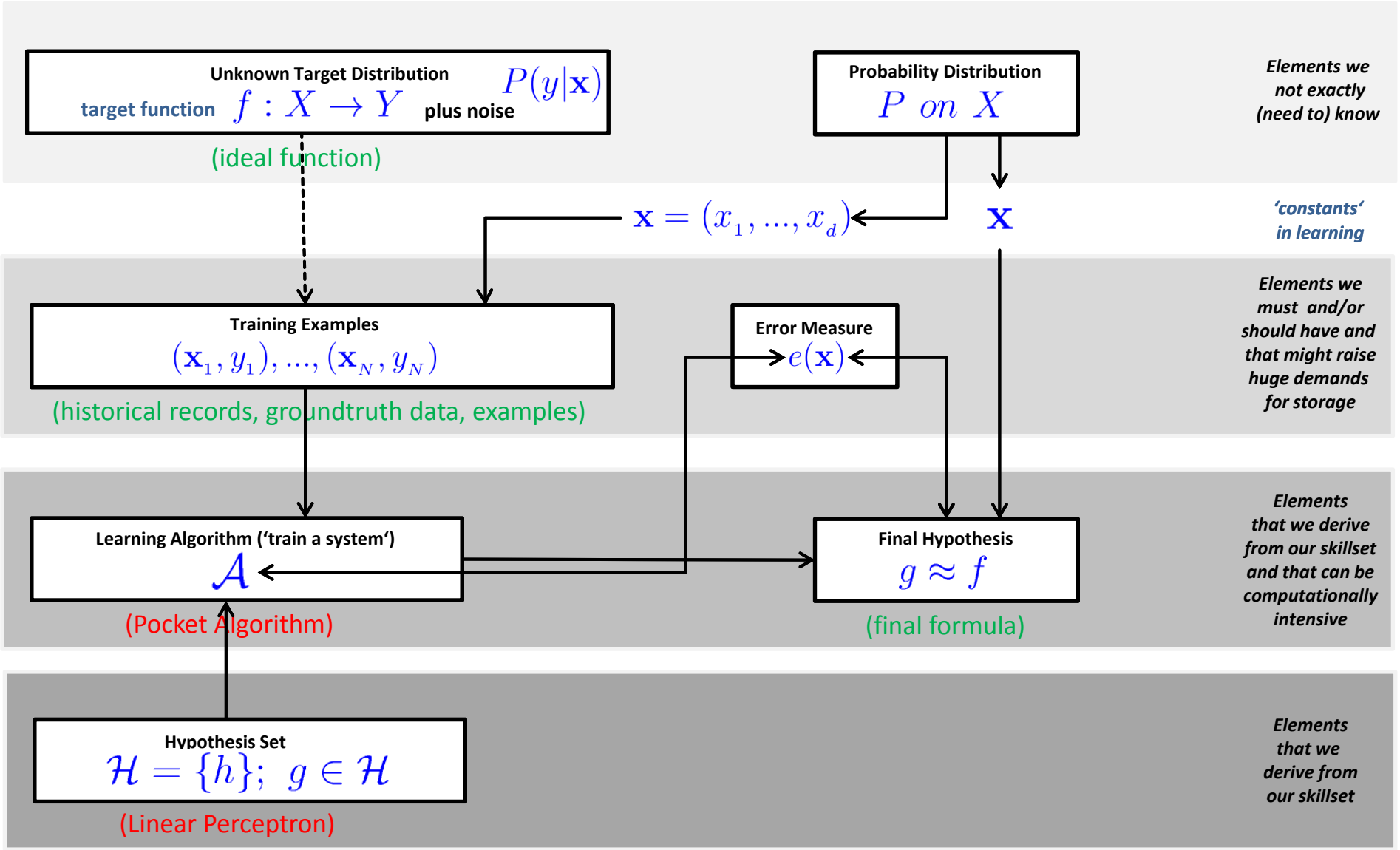
Smart Extension to PLA – Pocket Algorithm – Enough?

- When: If we believe there is a linear pattern to be detected
  - No assumption: can be non-linearly seperable data



- Basis is still the PLA
- Idea: Put the best solution so far 'in a pocket'
- Best means: Error measure checks per iterations
- Works with non-linearly seperable data
- Needs fixed iterations number (otherwise no convergence of algorithm)

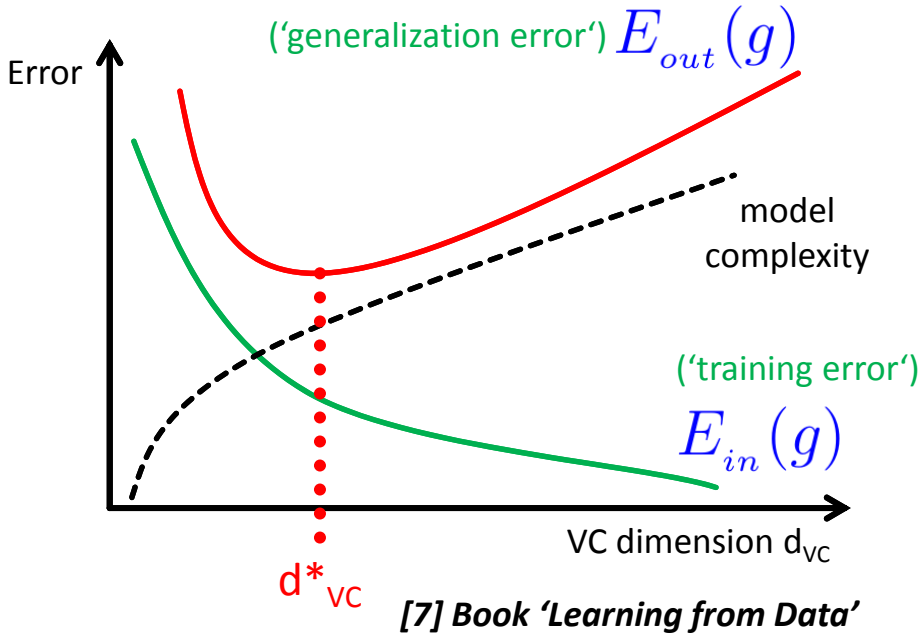
Solution Tools: Linear Perceptron Hypothesis Set & Pocket



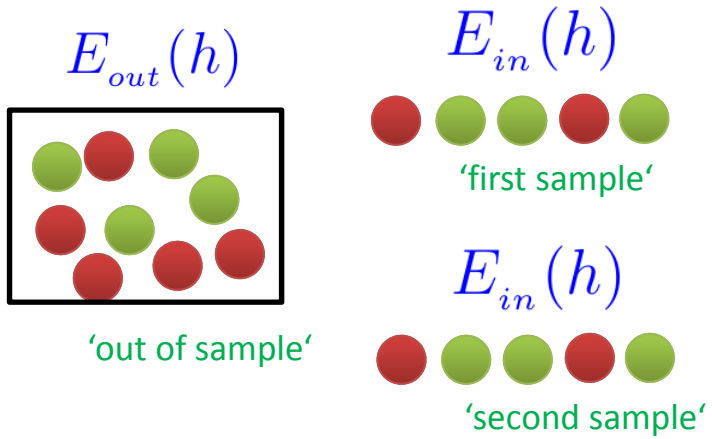
Towards Complexity of the Hypothesis Set – VC Dimension

- Vapnik-Chervonenkis (VC) Dimension over instance space X
  - VC dimension gets a 'generalization bound' on all possible target functions
  - Practice: think how much model parameters ('degrees of freedom')

Issue: unknown to 'compute' – VC solved this using the growth function on different samples



[7] Book 'Learning from Data'

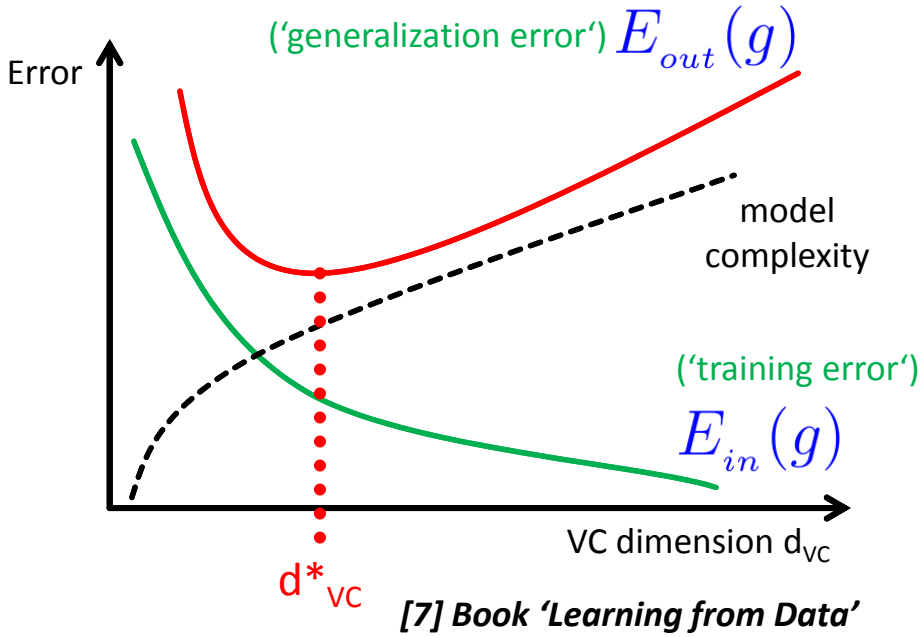


idea: 'first sample' frequency close to 'second sample' frequency

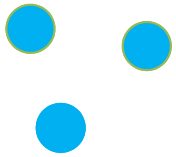
- **Complexity of Hypothesis set H can be measured by the Vapnik-Chervonenkis (VC) Dimension d_{VC}**
- **Ignoring the model complexity d_{VC} leads to situations where $E_{in}(g)$ gets down and $E_{out}(g)$ gets up**

VC Dimension – Example 2D Perceptron

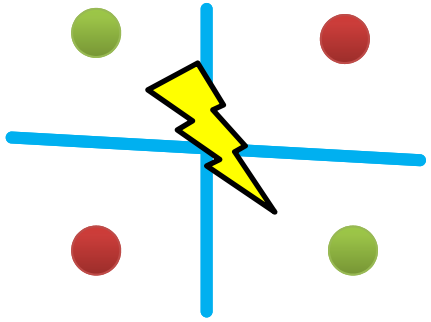
- Vapnik-Chervonenkis (VC) Dimension over instance space X
  - VC dimension gets a 'generalization bound' on all possible target functions
  - Practice: think how much model parameters ('degrees of freedom')
  - Practice: guides what model to use when receiving data to be analysed



[7] Book 'Learning from Data'



(VC dimension = 3, 2D Perceptron can shatter points)



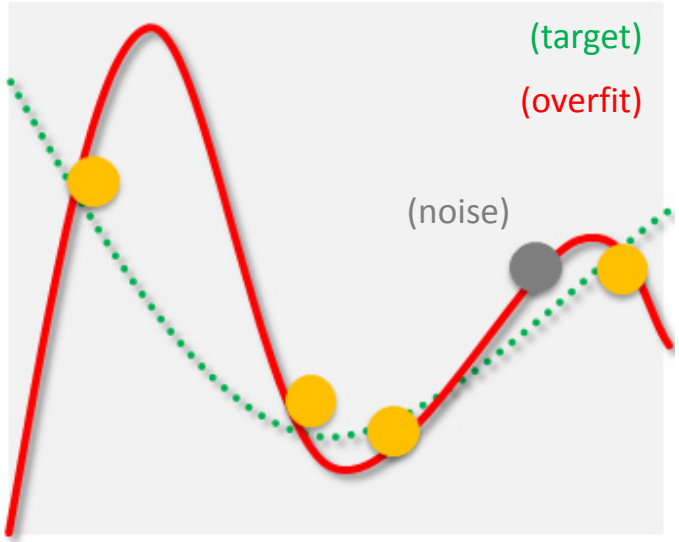
(breakpoint k = 4)

- **Utility of VC dimension gives 'rules of thumb' that ~10 * VC Dimension amount of data (N) samples are needed to 'start learning' (more better) with a given model in order to start generalizing well**

Challenge – Problem of Overfitting & Model Selection

- **Overfitting refers to fit the data too well – more than is warranted – thus may misguide the learning**
- **Overfitting is not just 'bad generalization' - e.g. the VC dimension covers noiseless & noise targets**
- **Theory of Regularization are approaches against overfitting and prevent it using different methods**

- Key problem: noise in the target function leads to overfitting
  - Effect: 'noisy target function' and its noise misguides the fit in learning
  - There is always 'some noise' in the data
  - Consequence: poor target function ('distribution') approximation
- Example: Target functions is second order polynomial (i.e. parabola)
  - Using a higher-order polynomial fit
  - Perfect fit: low $E_{in}(g)$, but large $E_{out}(g)$



(but simple polynomial works good enough)
('over': here meant as 4th order, a 3rd order would be better, 2nd best)

Initial Terminologies – Reviewed w.r.t. Model Decisions

- Target Function $f : X \rightarrow Y$ → Target Distribution $f : X \rightarrow Y$ plus noise $P(y|\mathbf{x})$
  - Ideal 'function' that 'explains' the data we want to learn
- Labelled Dataset (samples)
  - 'in-sample' data given to us $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
- Dataset Part One: Training set (training set is used to make some decisions for model…)
  - Used for training a machine learning algorithms
  - Result after using a training set: a trained system
- Dataset Part Two: Test set (testing set has not been used to make any decisions for model…)
  - Used for testing whether the trained system might work well
  - Result after using a test set: accuracy of the trained model
- Learning vs. Memorizing
  - The goal is to create a system that works well 'out of sample' (future data)

(Another set of data is needed not used in training but that is used for model selection & 'validate decisions')

Training and Testing – Reviewed w.r.t. Model Decisions

- Mathematical notations (cf. Lecture 5)
  - Testing follows: (hypothesis clear)
    $$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ >\epsilon\ ]\ <=\ 2\quad e^{-2\epsilon^2 N}$$
  - Training follows: (hypothesis search)
    $$\Pr\ [\ |\ E_{in}(g) - E_{out}(g)\ |\ >\epsilon\ ]\ <=\ 2Me^{-2\epsilon^2 N}$$
    (e.g. student exam training on examples to get E_{in} ‚down', then test via exam)
- Practice on 'training examples'
  - Create two disjoint datasets
  - One used for training only (aka training set)
  - Another used for testing only (aka test set)

Training Examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$$

(historical records, groundtruth data, examples)

- Training & Testing
  - Different phases in the learning process

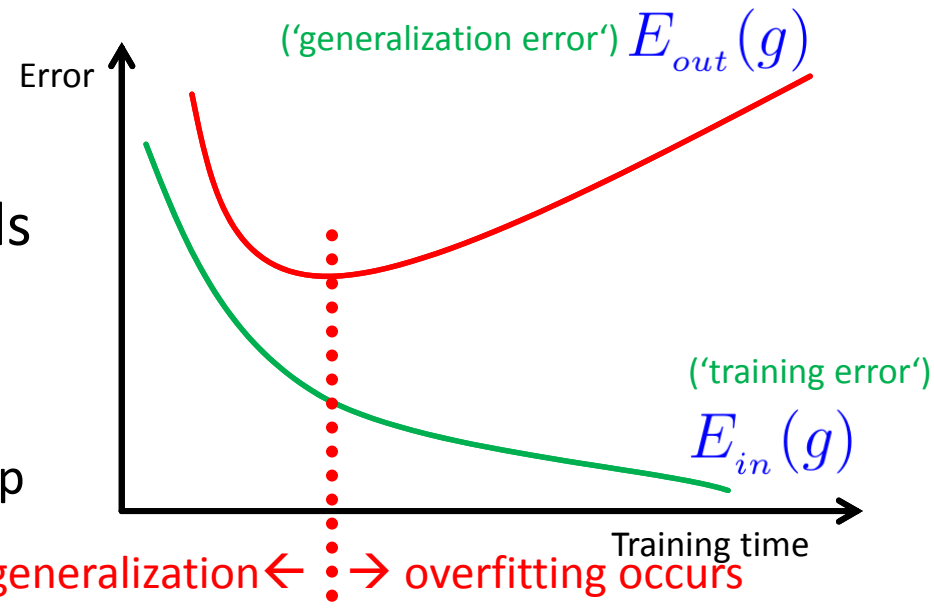
(Another phase in the creation of the whole model is needed where we take 'validated decisions about the model')

Problem of Overfitting – Clarifying Terms

- **A good model must have low training error (E_{in}) and low generalization error (E_{out})**
- **Model overfitting is if a model fits the data too well (E_{in}) with a poorer generalization error (E_{out}) than another model with a higher training error (E_{in})**

[2] Introduction to Data Mining

- Overfitting & Errors
  - $E_{in}(g)$ goes down
  - $E_{out}(g)$ goes up
- 'Bad generalization area' ends
  - Good to reduce $E_{in}(g)$
- 'Overfitting area' starts
  - Reducing $E_{in}(g)$ does not help
  - Reason 'fitting the noise'



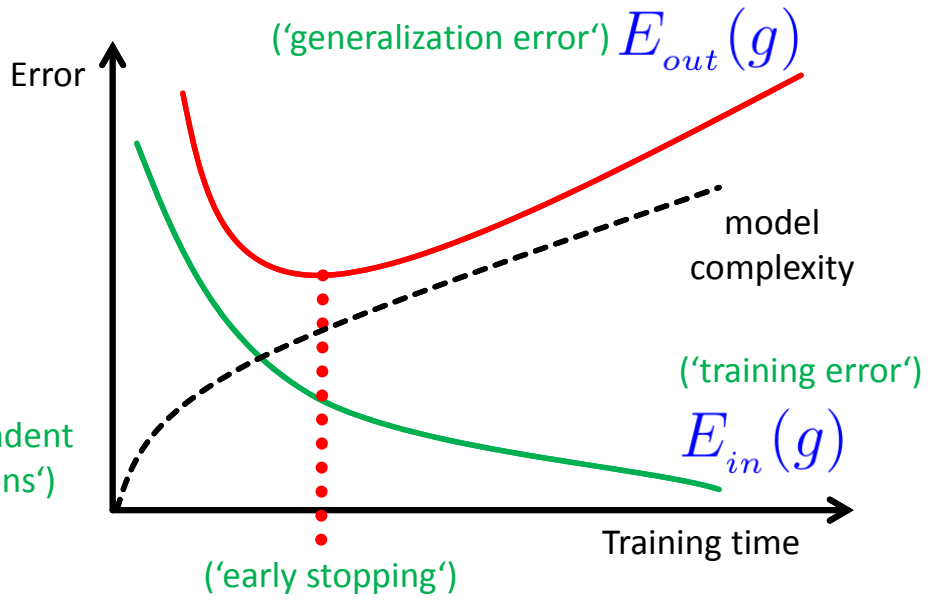
- **The two general approaches to prevent overfitting are (1) regularization and <u>(2) validation</u>**

(Decisions about the model are related to the problem of overfitting – need another method to 'select model well')

Problem of Overfitting – Model Relationships

- Review ‘overfitting situations‘
  - When comparing ‘various models‘ and related to ‘model complexity‘
  - Different models are used, e.g. 2nd and 4th order polynomial
  - Same model is used with e.g. two different instances (e.g. two neural networks but with different parameters)
- Intuitive solution
  - Detect when it happens
  - ‘Early stopping regularization term‘ to stop the training
  - Early stopping method (later)

(‘model complexity measure: the VC analysis was independent of a specific target function – bound for all target functions‘)



- **‘Early stopping‘ approach is part of the theory of regularization, but based on validation methods**

Problem of Overfitting – Noise Term Revisited

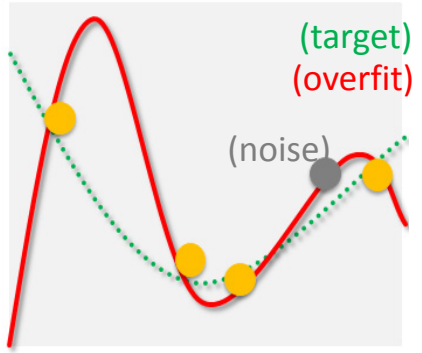
- ‘(Noisy) Target function‘ is not a (deterministic) function
  - Getting with ‘same x in‘ the ‘same y out‘ is not always given in practice
  - Idea: Use a ‘target distribution‘ instead of ‘target function‘

Unknown Target Distribution

target function $f : X \rightarrow Y$ plus noise $P(y|\mathbf{x})$

(ideal function)

- **Fitting some noise in the data is the basic reason for overfitting and harms the learning process**
- **Big datasets tend to have more noise in the data so the overfitting problem might occur even more intense**



(‘function view‘)

‘shift the view’

(‘# data view‘)

‘# samples‘

- ‘Different types of some noise‘ in data
  - Key to understand overfitting & preventing it
  - ‘Shift of view‘: refinement of noise term
  - Learning from data: ‘matching properties of # data‘

Problem of Overfitting – Stochastic Noise

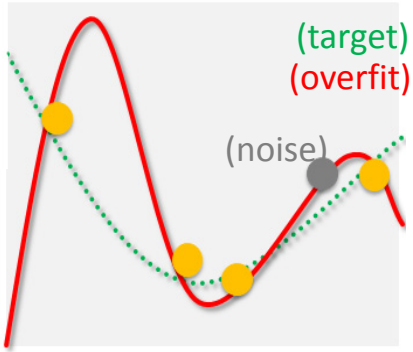
- Stoachastic noise is a part ‘on top of‘ each learnable function
  - Noise in the data that can not be captured and thus not modelled by f
  - Random noise : aka ‘non-deterministic noise‘
  - Conventional understanding established early in this course
  - Finding a ‘non-existing pattern in noise not feasible in learning‘

Unknown Target Distribution

target function $f : X \rightarrow Y$ plus noise $P(y|\mathbf{x})$

(ideal function)

- Practice Example
  - Random fluctuations and/or measurement errors in data (cf. Lecture 1, PANGAEA)
  - Fitting a pattern that not exists ‘out-of-sample‘
  - Puts learning progress ‘off-track‘ and ‘away from f‘

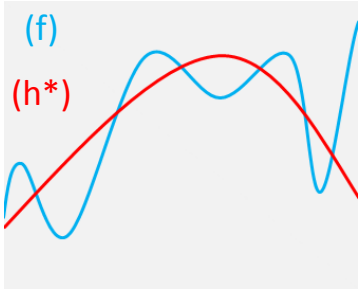


- **Stochastic noise here means noise that can‘t be captured, because it‘s just pure ‘noise as is‘ (nothing to look for) – aka no pattern in the data to understand or to learn from**

Problem of Overfitting – Deterministic Noise

- Part of target function f that H can not capture: $f(\mathbf{x}) - h^*(\mathbf{x})$
  - Hypothesis set H is limited so best h* can not fully approximate f
  - h* approximates f, but fails to pick certain parts of the target f
  - 'Behaves like noise', existing even if data is 'stochastic noiseless'
- Different 'type of noise' than stochastic noise
  - Deterministic noise depends on $\mathcal{H}$ (determines how much more can be captured by h*)
  - E.g. same f, and more sophisticated $\mathcal{H}$: noise is smaller (stochastic noise remains the same, nothing can capture it)
  - Fixed for a given $\mathbf{x}$, clearly measurable (stochastic noise may vary for values of $\mathbf{x}$)

(learning deterministic noise is outside the ability to learn for a given h*)



- **Deterministic noise here means noise that can't be captured, because it is a limited model (out of the league of this particular model), e.g. 'learning with a toddler statistical learning theory'**

Problem of Overfitting – Impacts on Learning

- **The higher the degree of the polynomial (cf. model complexity), the more degrees of freedom are existing and thus the more capacity exists to overfit the training data**

- Understanding deterministic noise & target complexity
  - Increasing target complexity increases deterministic noise (at some level)
  - Increasing the number of data N decreases the deterministic noise
- Finite N case: $\mathcal{H}$ tries to fit the noise
  - Fitting the noise straightforward (e.g. Perceptron Learning Algorithm)
  - Stochastic (in data) and deterministic (simple model) noise will be part of it
- Two 'solution methods' for avoiding overfitting
  - Regularization: 'Putting the brakes in learning', e.g. early stopping (more theoretical, hence 'theory of regularization')
  - Validation: 'Checking the bottom line', e.g. other hints for out-of-sample (more practical, methods on data that provides 'hints')

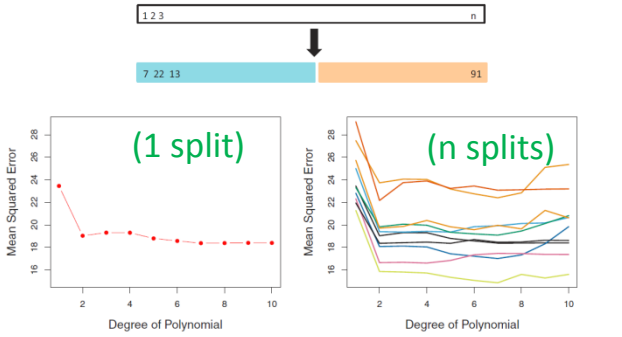
Validation & Model Selection – Terminology

- **The ‘Validation technique‘ should be used in all machine learning or data mining approaches**
- **Model assessment is the process of evaluating a models performance**
- **Model selection is the process of selecting the proper level of flexibility for a model**

modified from [3] ‘An Introduction to Statistical Learning’

- ‘Training error‘
  - Calculated when learning from data (i.e. dedicated training set)
- ‘Test error’
  - Average error resulting from using the model with ‘new/unseen data‘
  - ‘new/unseen data‘ was not used in training (i.e. dedicated test set)
  - In many practical situations, a dedicated test set is not really available
- ‘Validation Set‘
  - Split data into training & validation set
- ‘Variance‘ & ‘Variability‘
  - Result in different random splits (right)

(split creates a two subsets of comparable size)



Validation Technique – Formalization & Goal

- **Validation is a very important technique to estimate the out-of-sample performance of a model**
- **Main utility of regularization & validation is to control or avoid overfitting via model selection**

- Regularization & Validation
  - Approach: introduce a 'overfit penalty' that relates to model complexity
  - Problem: Not accurate values: 'better smooth functions'

(regularization uses a term that captures the overfit penalty)

$$E_{out}(h) = E_{in}(h) + \textbf{overfit penalty}$$

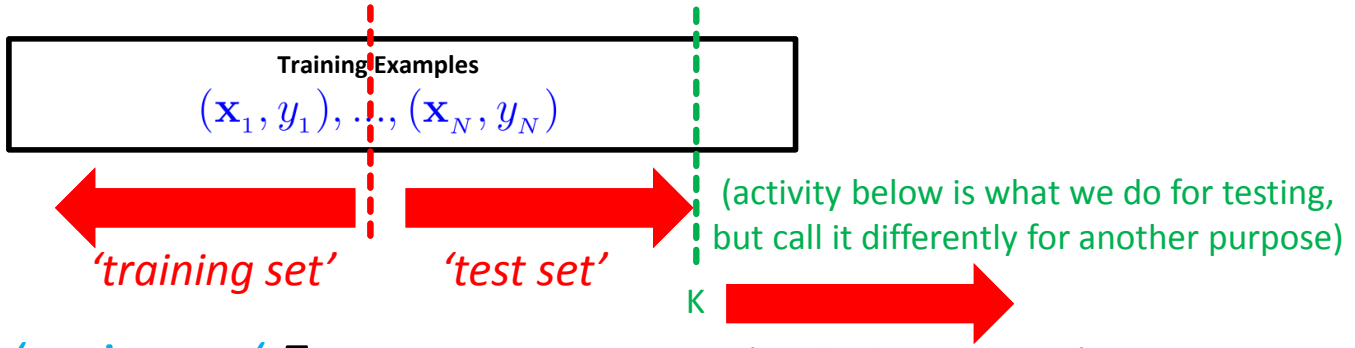
(minimize both to be better proxy for E_{out})

(validation estimates this quantity)

(regularization estimates this quantity)

- Validation (measuring E_{out} is not possible as this is an unknown quantity, another quantity is needed that is measurable that at least estimates it)
  - Goal 'estimate the out-of-sample error' (establish a quantity known as validation error)
  - Distinct activity from training and testing (testing also tries to estimate the E_{out})

Validation Technique – Pick one point & Estimate E_{out}



- Understanding 'estimate' E_{out}
  - On one out-of-sample point $(\mathbf{x}, y)$ the error is $e(h(\mathbf{x}), y)$
  - E.g. use squared error: $e(h(\mathbf{x}), f(\mathbf{x})) = (h(\mathbf{x}) - f(\mathbf{x}))^2$
    $e(h(\mathbf{x}), y) = (h(\mathbf{x}) - y)^2$
  - Use this quantity as estimate for E_{out} (poor estimate)
  - Term 'expected value' to formalize (probability theory)

(Taking into account the theory of Lecture 1 with probability distribution on X etc.)

Probability Distribution
P on X

$\mathbf{x} = (x_1, \ldots, x_d)$

(aka 'random variable')

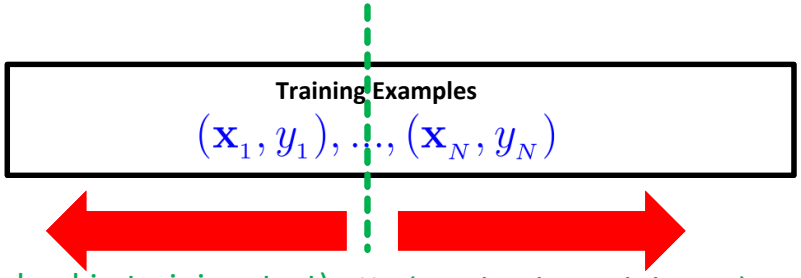
$\mathbb{E}[e(h(\mathbf{x}), y)] = E_{out}(h)$ (aka the long-run average value of repetitions of the experiment)

(one point as unbiased estimate of E_{out} that can have a high variance leads to bad generalization)

Validation Technique – Validation Set

- **Validation set consists of data that has been not used in training to estimate true out-of-sample**
- **Rule of thumb from practice is to take 20% (1/5) for validation of the learning model**

- Solution for high variance in expected values $\mathbb{E}[e(h(\mathbf{x}), y)] = E_{out}(h)$
  - Take a 'whole set' instead of just one point $(\mathbf{x}, y)$ for validation



(we need points not used in training
to estimate the out-of-sample performance)

(we do the same approach with the
testing set, but here different purpose)

  - Idea: K data points for validation

$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)$ (validation set)

$$E_{val}(h) = \frac{1}{K} \sum_{k=1}^{K} e(h(\mathbf{x})_k, y_k)$$ (validation error)

  - Expected value to 'measure' the out-of-sample error
  - 'Reliable estimate' if K is large

(expected values averaged over set)

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

(on rarely used validation set,
otherwise data gets contaminated)

(this gives a much better (lower) variance than on a single point given K is large)

Validation Technique – Model Selection Process

- **Model selection is choosing (a) different types of models or (b) parameter values inside models**
- **Model selection takes advantage of the validation error in order to decide → 'pick the best'**

Hypothesis Set

$\mathcal{H} = \{h\};\ g \in \mathcal{H}$

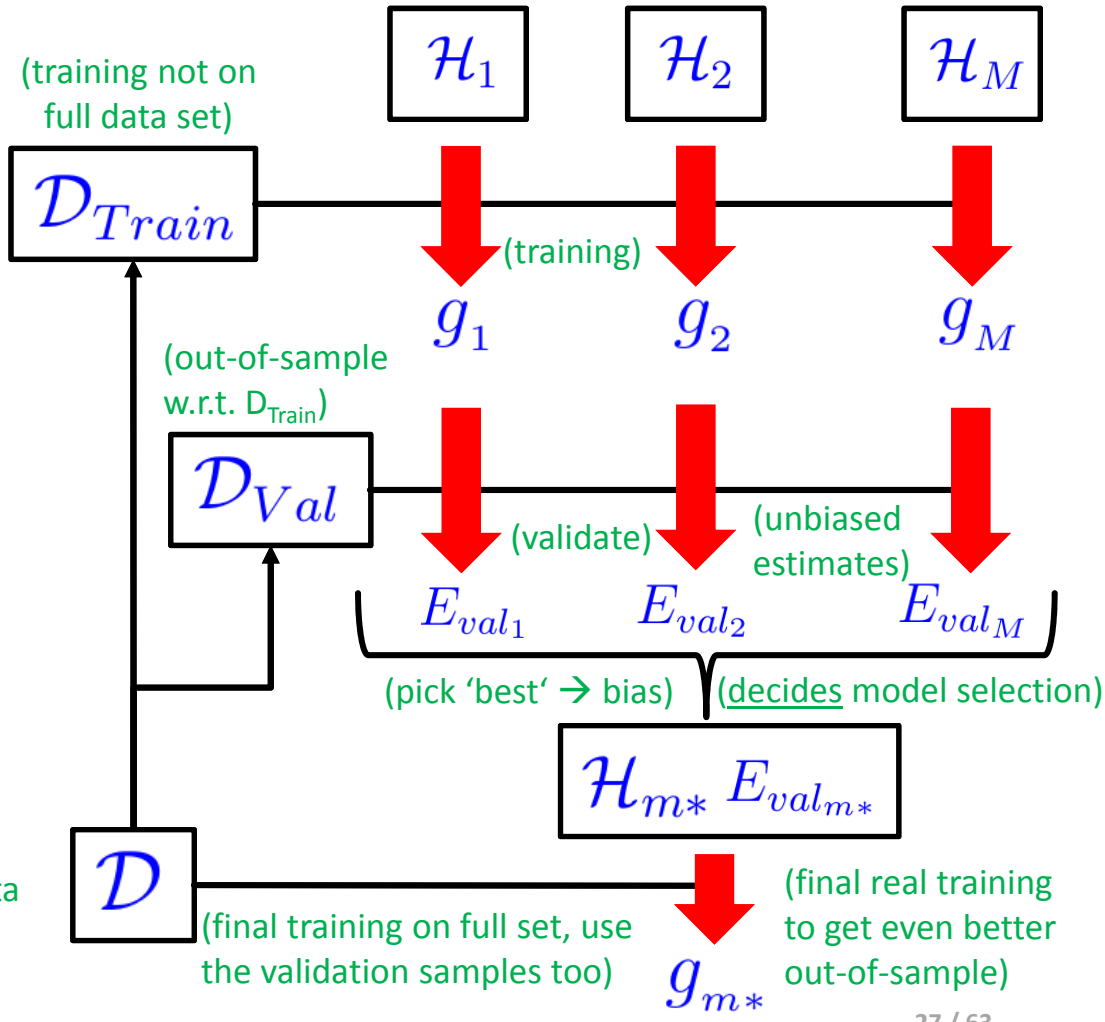
(set of candidate formulas across models)

- M models (cf. Lecture 1)
  - Use validation error to perform select decisions
- Careful consideration:
  - 'Picked means decided' hypothesis has already bias (→ contamination)
  - Using $\mathcal{D}_{Val}$ M times

Final Hypothesis

$g_{m*} \approx f$

(test this on unseen data good, but depends on availability in practice)

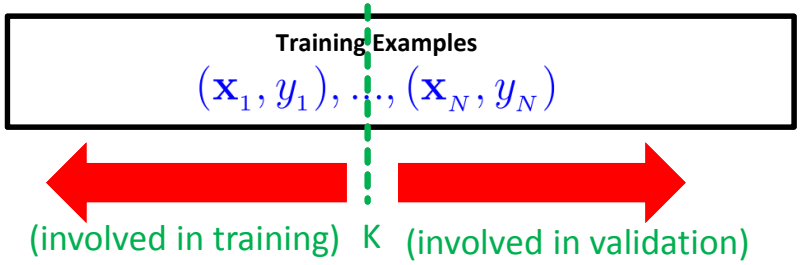


Validation Technique – Cross-Validation – Trick

- **Cross-validation the technique of choice in practical situations to perform model selection**
- **Different techniques exist for cross-validation such as leave-one-out, leave-more-out**

(every time a data point is used for validation it is taken away from training)

- Goal (validation data not given on top of training data)
  - Target issue 'choosing K' out of N
  - Issue: K needs to be small & large
  - (cf. Lecture 1 feasibility of learning)



(conflicting requirements on K)

(chain of reasoning so far)

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

(idea: is there a solution over time?)

(small K for large N – K training delivering good out-of-sample performance)

(large K for validation delivering a good estimate for out-of-sample performance)

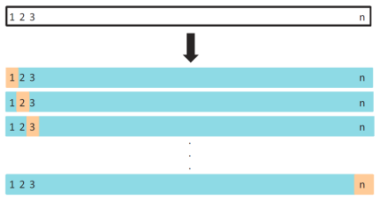
- Apply trick: repeat the number of trainings on different subsets
  - Train multiple times using e.g. leave-one-out or leave-more-out (practice)

- **Cross-validation 'trick' achieves to use N points for training and N points for validation (big gain!)**

Validation Technique – Cross-Validation – Leave-one-out

- Simplest form of cross-validation
  - Use N – 1 data points for training
  - Potential issue: only 1 data point for validation (bad estimate for E_{out})
  - Creates a very 'small validation' set, but very 'large training' set

(split data NOT just in two subsets of comparable size)



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

(reduced dataset, but as training set very close to N missing just one)

(one data point left out for validation)

- Final hypothesis to be 'selected' after training with $\mathcal{D}_n$ is g_n^-

(not trained on full set & depends on point left out)

(check error on the point 'left out' → out-of-sample)

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

(the hypothesis was trained not involving this point)

(validate hypothesis on that data point taken out)

(one point in validation set brings bad estimate for E_{out})

- Apply 'resampling' trick:
  Repeat for different n with $\mathcal{D}_n$

(obtains different hypothesis g_1-, g_2-,…., but all have in common to be obtained by being trained on N – 1 points)

$$E_{CV} = \frac{1}{N} \sum_{n=1}^{N} e_n$$

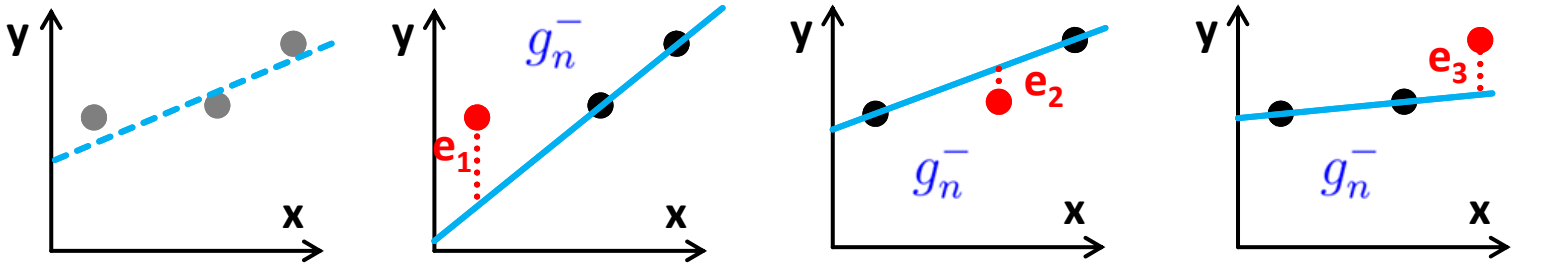
(works well with increasing N)

(cross-validation error on validation set with N points)

Validation Technique – Cross-Validation Error Example

- Example: Create a linear model
  - Assuming there is noise in the target function
  - Cross-validation: evaluate out-of-sample error to choose a model (later)

(red points are validation sets in each run) (black points are training sets in each run)



(the full dataset = 3 points) (n = left point out) (n = middle point out) (n = right point out)

$$E_{CV} = \frac{1}{N}\sum_{n=1}^{N} e_n$$

(simply compute average of all errors, e.g. using squared distance)

$$E_{CV} = \frac{1}{3}(e_1 + e_2 + e_3)$$

(cross-validation error as indication of how well 'the linear model' fits the data → out-of-sample)

(impact on N = small (3) is enormous, but if N = large average works very well)

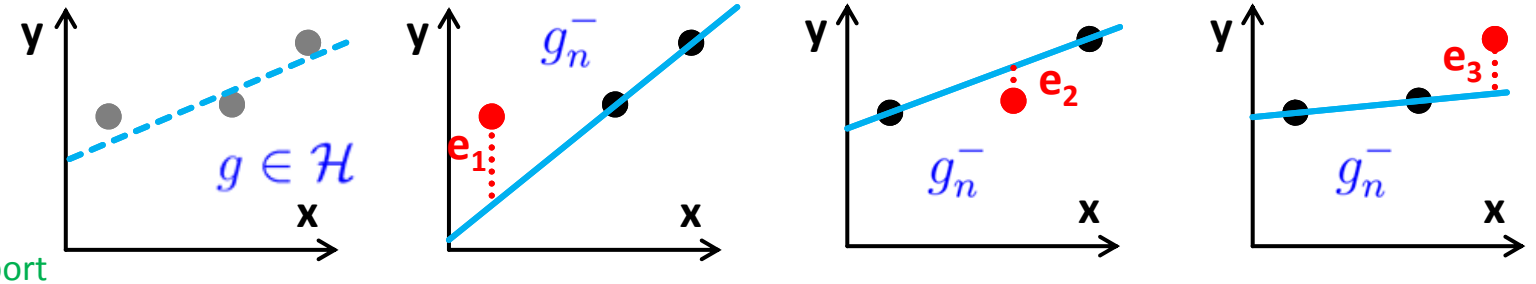
- **Cross-validation is a 'resampling method' that obtains more information than 'fitting model once'**
- **Compute cross-validation error is possible (via 'in-sample') & a systematic way for model selection**

Validation Technique – Cross-Validation & Model Selection

- Model selection: Perform a ‘decision‘
  - Cross-validation: evaluate out-of-sample error to choose a model (avoiding e.g. heuristics)
  - Example: Decide whether Linear Model or Constant Model is better

Hypothesis Set
$\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(set of candidate formulas)

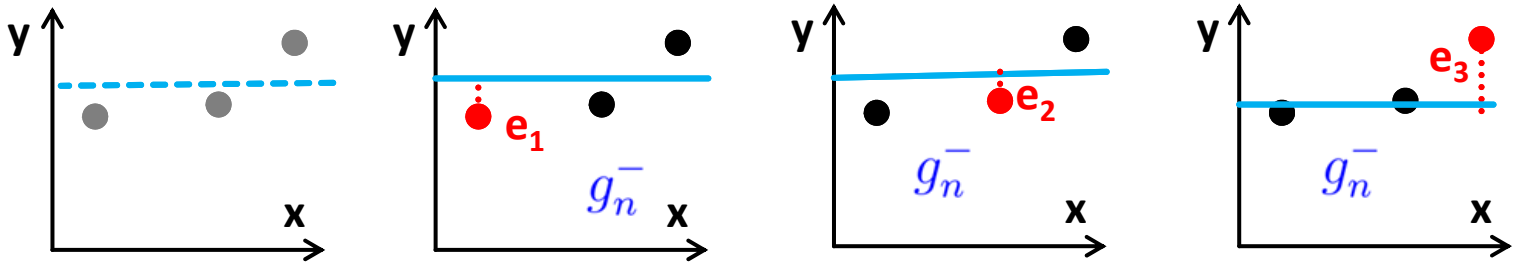
$\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(LINEAR MODEL)



(decision support required, think N huge)

(use the obtained cross-validation error for both models as a ‘grade for each model‘)
(use this ‘grade‘ and compare it for model selection, e.g. constant model wins, less squared error)

$\mathcal{H} = \{h\};\ g \in \mathcal{H}$
(CONSTANT MODEL)



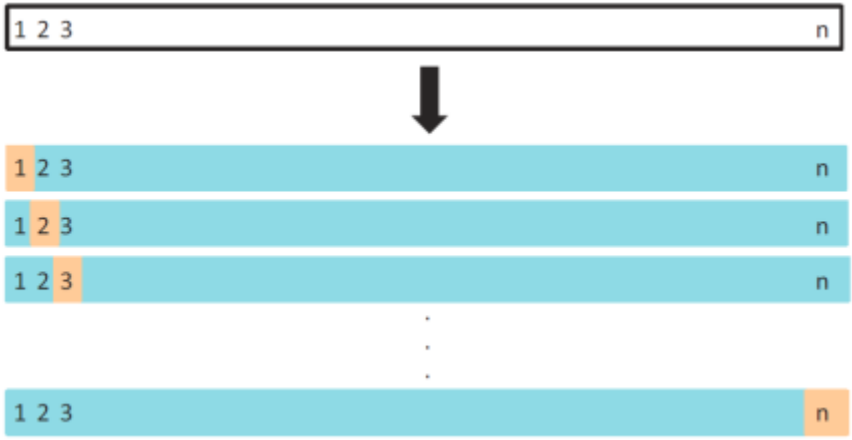
- **Main utility of cross-validation is model selection supporting a decision to choose a model**

Validation Technique – Cross-Validation – K-Fold Approach

(split creates a two subsets of comparable size)

(random strategy, works not particularly well)

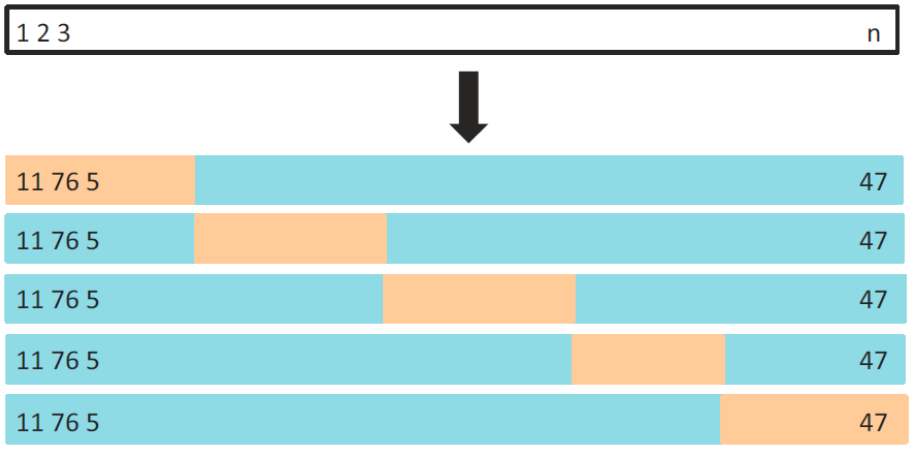
Leave-One-Out Cross-Validation (LOOCV) Example



(picking strategy, works well but possible long computing)

pick one point for validation
resulting in possible large
sets when number of points are high

5-fold Cross-Validation Example



(picking strategy, works well and reduces computing)

A set of data points is randomly split into
k non-overlapping groups ('k-folds')
of approximately equal size

Recommendation in Practice

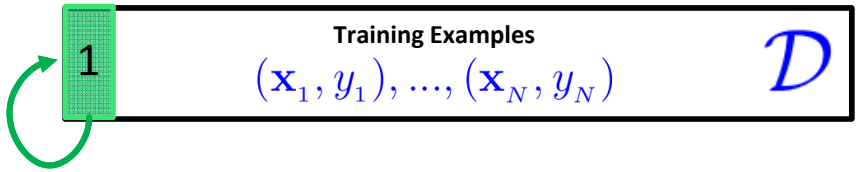
[3] 'An Introduction to Statistical Learning'

Validation Technique – Cross-Validation – Leave-more-out

- **10-fold cross validation is mostly applied in practical problems by setting K = N/10 for real data**
- **Having N/K training sessions on N – K points each leads to long runtimes (→ use parallelization)**

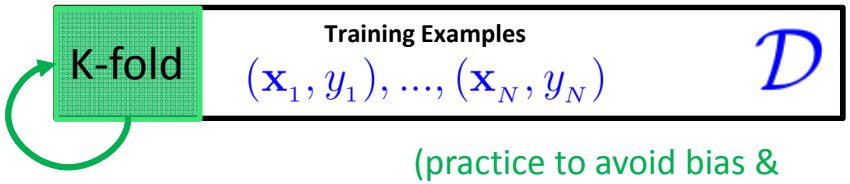
- Leave-one-out
  - N training sessions on N – 1 points each time

(leave 1 point out at each run → many runs)



- Leave-more-out
  - Break data into number of folds
  - N/K training sessions on N – K points each time (fewer training sessions than above)

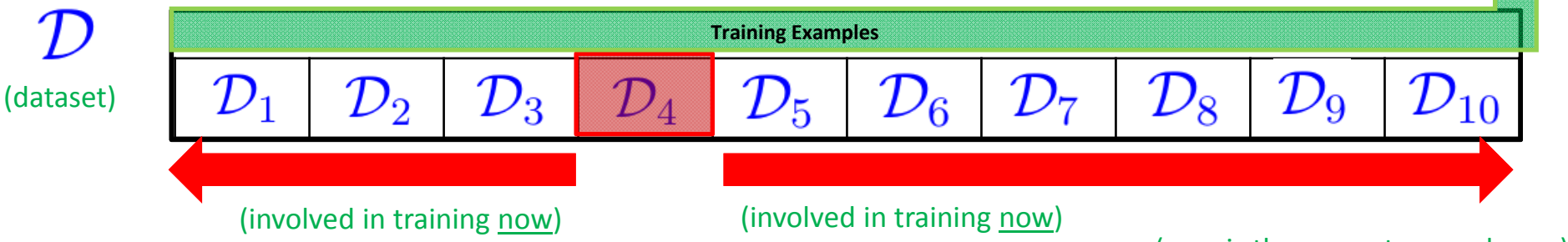
(generalization to leave k points out at each run)



(practice to avoid bias & contamination: some rest for test as 'unseen data')

- Example: '10-fold cross-valdation' with K = N/10 multiple times (N/K)

(use 1/10 for validation, use 9/10 for training, then another 1/10 … N/K times)

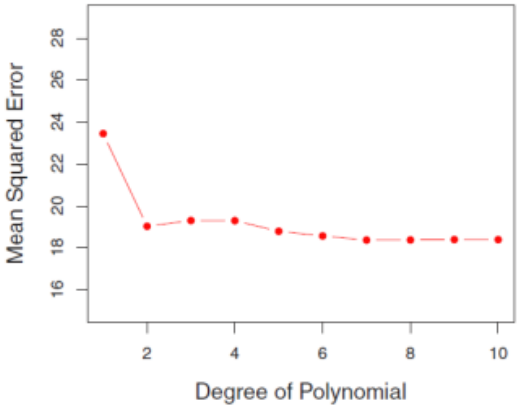


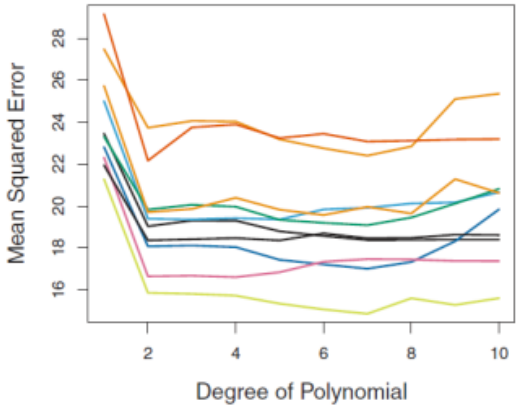
(involved in training now) (involved in training now)

(now is the current example run)

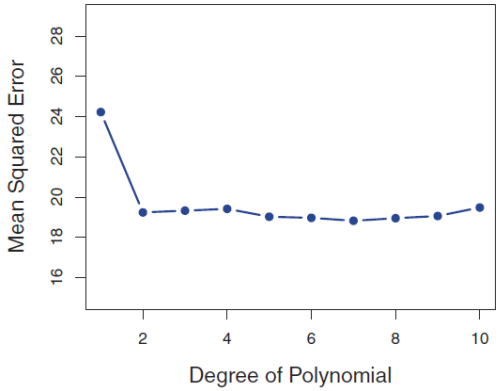
Validation Technique – 10 fold Cross-Validation Example

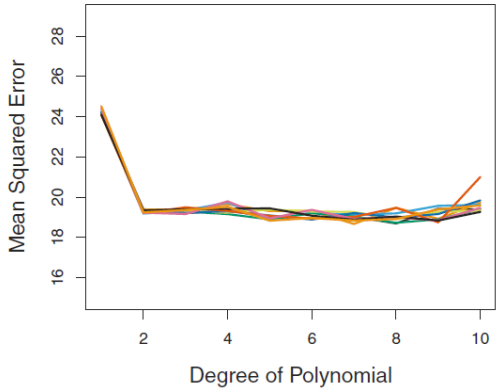
- 10 times resampling
  - Validation set with 10 x 2 comparable sizes
  - ‘Random splits‘
  - High variability/variance





- 10-fold cross validation
  - Validation set with 10-fold x 2 strategy
  - No ‘random splits‘
  - Lower variability/variance





modified from [3] ‘An Introduction to Statistical Learning’

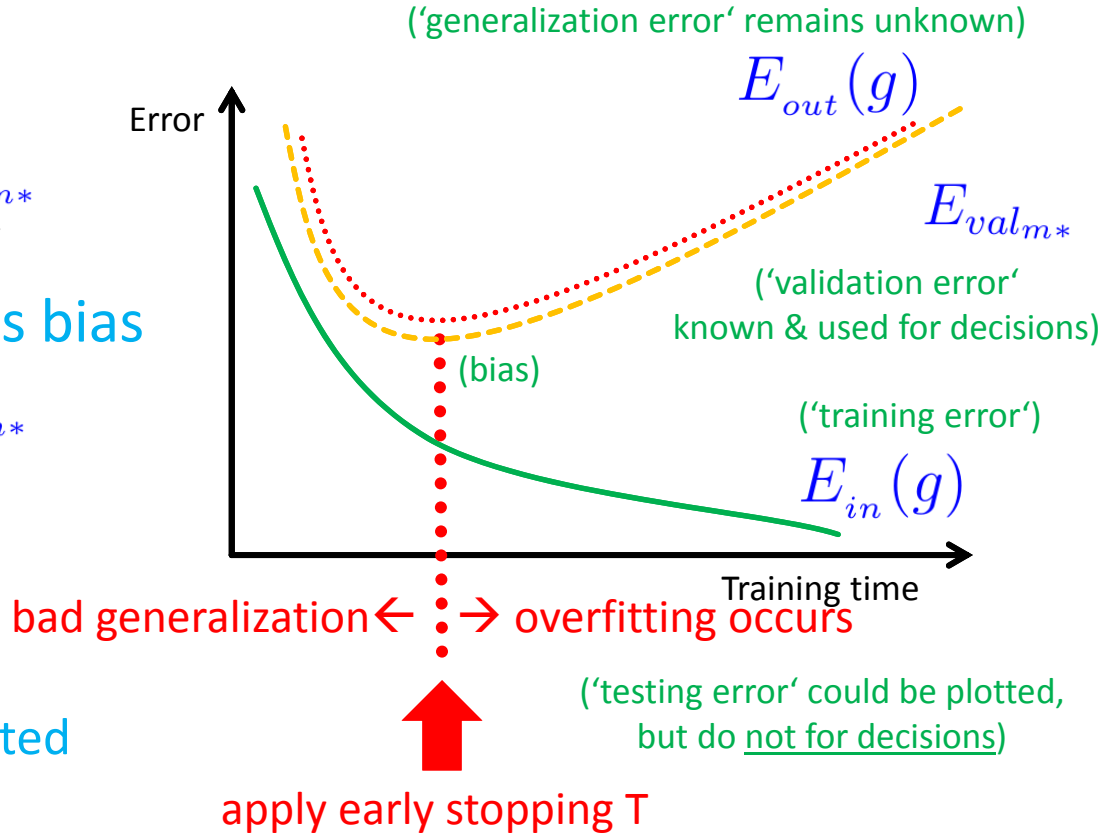
Model Performance – Validation Enables Early Stopping (1)

- Problem of overfitting
  - Issue is that $E_{out}(g)$ is unknown to perform 'early stopping'
  - Apply validation = 'perform decision to make a choice' based on $\mathcal{D}_{Val}$

$\mathcal{D}_{Val}$ (out-of-sample w.r.t. D_{Train})

  - Use validation error $E_{val_{m*}}$ to perform early stopping

- Validation decision brings bias
  - When the estimate $E_{val_{m*}}$ of $E_{out}(g)$ affects the learning process decision
  - Optimistic bias impact brings accuracy higher than in reality (cf. Associated use case with testset)



Model Performance – Validation Enables Early Stopping (2)

- ‘Bias‘ reviewed as ‘data contamination‘
  - Training set is biased and contaminated (i.e. ‘used for train model change‘)
  - Test set is unbiased and clean (i.e. ‘waiting to be used in the final end‘)
  - Validation set has an optimistic bias (i.s. ‘use in model selection decisions‘)

(‘slightly contaminated since only few choices‘)

$$E_{val_1} \quad E_{val_2} \quad E_{val_M}$$

(pick ‘best‘ → bias) (decides model selection)

$$\mathcal{H}_{m*} \; E_{val_{m*}}$$

(reasoning of bias relates to the probability and estimated value of validation errors since ‘one is picked‘ as the minimum of all)

(e is a min function of E_{val1}, E_{val2}, etc.)

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

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

(aka the long-run average value of repetitions of the experiment)

(cf. picking the ‘best time‘ in early stopping, also brings optimistic bias since minimum on model creation)

- **Optimistic bias means that there is ‘a belief‘ that the error is smaller as it is actually going to be**
- **Optimistic bias is minor and thus accepted in learning, but perform reporting with unbiased testset**
- **Important in validation is that the validation set stays only ‘slightly contaminated‘ (few choices)**
- **In practice several validation sets can be used for n parameter choices to keep reliable estimate**

Exercises – 10-fold Cross-Validation Runs Rome

JURECA System – SSH Login

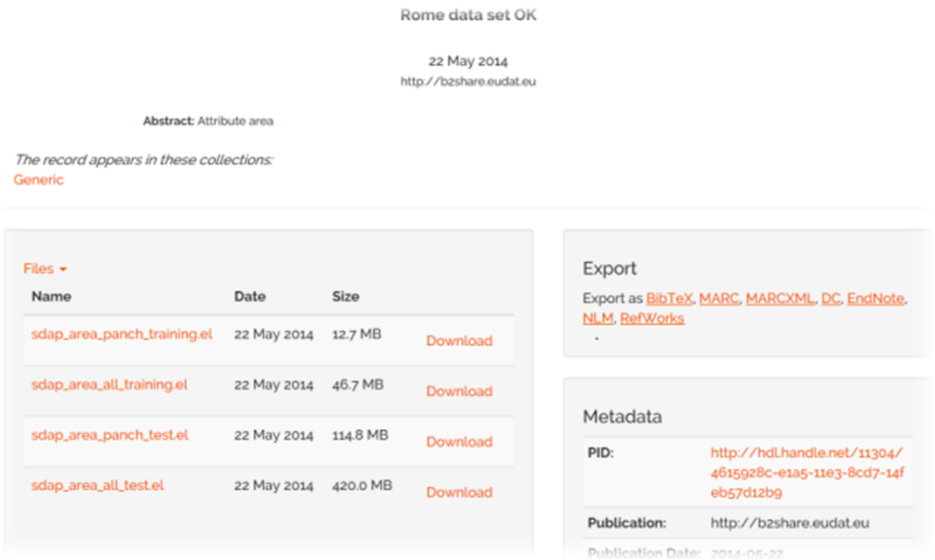
- Use your account train004 - train050
- Windows: use mobaxterm or putty
- UNIX: ssh trainXYZ@jureca.fz-juelich.de
- Example

```
adminuser@linux-8djg:~> ssh train001@jureca.fz-juelich.de
Warning: the ECDSA host key for 'jureca.fz-juelich.de' differs from the key for the IP address '134.94.33.
9'
Offending key for IP in /home/adminuser/.ssh/known_hosts:12
Matching host key in /home/adminuser/.ssh/known_hosts:19
Are you sure you want to continue connecting (yes/no)? yes
]Last login: Mon Aug 21 14:29:03 2017 from zam2036.zam.kfa-juelich.de
*******************************************************************************
*                            Welcome to JURECA                                *
*                                                                             *
*  Information about the system, latest changes, user documentation and FAQs: *
*                  http://www.fz-juelich.de/ias/jsc/jureca                    *
*******************************************************************************
*                           ### Known Issues ###                              *
*                                                                             *
* An up-to-date list of known issues on the system is maintained at           *
*               http://www.fz-juelich.de/ias/jsc/jureca-known-issues          *
* Open issues:                                                                *
*   - Intel compiler error with std::valarray and                             *
*     optimized headers, added 2016-03-20                                     *
```

➢ **Remember to use your own trainXYZ account in order to login to the JURECA system**

Rome Remote Sensing Dataset (cf. Lecture 6)

- Data is already available in the tutorial directory



(persistent handle link for publication into papers)

B2SHARE
Store and Share Research Data

[4] Rome Image dataset

```
[train001@jrl12 rome]$ pwd
/homea/hpclab/train001/data/rome
[train001@jrl12 rome]$ ls -al
total 580256
drwxr-xr-x 2 train001 hpclab       512 Jan 14 21:52 .
drwxr-xr-x 6 train001 hpclab       512 Jan 14 21:47 ..
-rw-r--r-- 1 train001 hpclab 419974873 Dec 22  2016 sdap_area_all_test.el
-rw-r--r-- 1 train001 hpclab  46652874 Dec 22  2016 sdap_area_all_training.el
-rw-r--r-- 1 train001 hpclab 114763982 Dec 22  2016 sdap_area_panch_test.el
-rw-r--r-- 1 train001 hpclab  12745692 Dec 22  2016 sdap_area_panch_training.el
```

HPC Environment – Modules Revisited

- Module environment tool
  - Avoids to manually setup environment information for every application
  - Simplifies shell initialization and lets users easily modify their environment
- Module avail
  - Lists all available modules on the HPC system (e.g. compilers, MPI, etc.)
- Module spider
  - Find modules in the installed set of modules and more information
- Module load → needed before piSVM run
  - Loads particular modules into the current work environment, E.g.:

```
[train001@jrl12 ~]$ module load Intel
[train001@jrl12 ~]$ module load IntelMPI
```

Parallel & Scalable PiSVM – Parameters

```
[train001@j3l02 pisvm-1.2.1]$ ./pisvm-train
Usage: svm-train [options] training_set_file [model_file]
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/k)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 40)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
-o n: max. size of working set
-q n: max. number of new variables entering working set
flags:
-D: Assume the feature vectors are dense (default: sparse)
```

- **n-fold cross-validation is an estimate of the out-of-sample performance**
- **As it is 'just' an estimate the n-fold cross-validation run will not create any model, but gives a reliable estimate for unseen data**

10-fold cross-validation Rome on JURECA – Job Script

- Use Rome Dataset with paralle & scalable piSVM tool
  - Parameters are equal to the serial libsvm and some additional parameters for paralellization

```
#!/bin/bash -x
#SBATCH--nodes=2
#SBATCH--ntasks=48
#SBATCH--ntasks-per-node=24
#SBATCH--output=mpi-out.%j
#SBATCH--error=mpi-err.%j
#SBATCH--time=01:00:00
#SBATCH--partition=batch
#SBATCH--mail-user=m.riedel@fz-juelich.de
#SBATCH--mail-type=ALL
#SBATCH--job-name=10x-cv-rome-rbf-2-48-24

### location executable
PISVM=/homea/hpclab/train001/tools/pisvm-1.2.1/pisvm-train

### location data
TRAINDATA=/homea/hpclab/train001/data/rome/sdap_area_all_training.el

### submit
srun $PISVM -v 10 -D -o 1024 -q 512 -c 100 -g 8 -t 2 -m 1024 -s 0 $TRAINDATA
```

- **Note the tutorial reservation with –reservation=ml-hpc-3 just valid for today on JURECA**

10-fold Cross-Validation Rome on JURECA – Outcome

- The output of the 10-fold cross-validation run is not a model
  - Used to estimate the out-of-sample performance & model selection
  - Note: we are not using the testset yet – just the training set

```
[train001@jrl06 pisvm-1.2.1]$ tail mpi-out.4642590
I/O time = 2.96
I/O time = 2.97
Cross Validation Accuracy = 6.17988%
Cross Validation Accuracy = 6.17988%
I/O time = 2.97
I/O time = 2.96
Cross Validation Accuracy = 6.17988%
I/O time = 2.98
Cross Validation Accuracy = 97.9044%
I/O time = 2.98
```

- The output of the testing/predicting phase is accuracy
  - Accuracy measurement of model performance (cf. Lecture 1)
  - The job output file consists of a couple of lines:

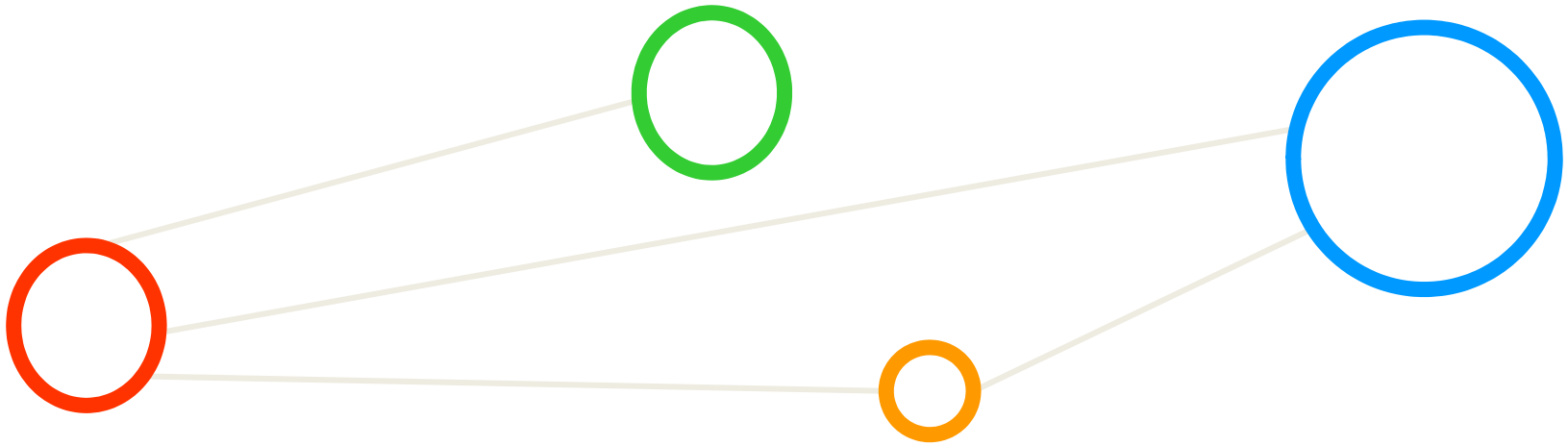
```
[train001@j3l02 pisvm-1.2.1]$ more mpi-out.15244544
Accuracy = 98.0558% (684291/697859) (classification)
Mean squared error = 0.195456 (regression)
Squared correlation coefficient = 0.968742 (regression)
```

[Video] Contamination of Data: Training, Testing, Validation

(relative high-level but captures the essence of unseen data and differences between testing & validation)

[5] YouTube Video, 'Machine Learning : Model Selection & Cross Validation'

Regularization

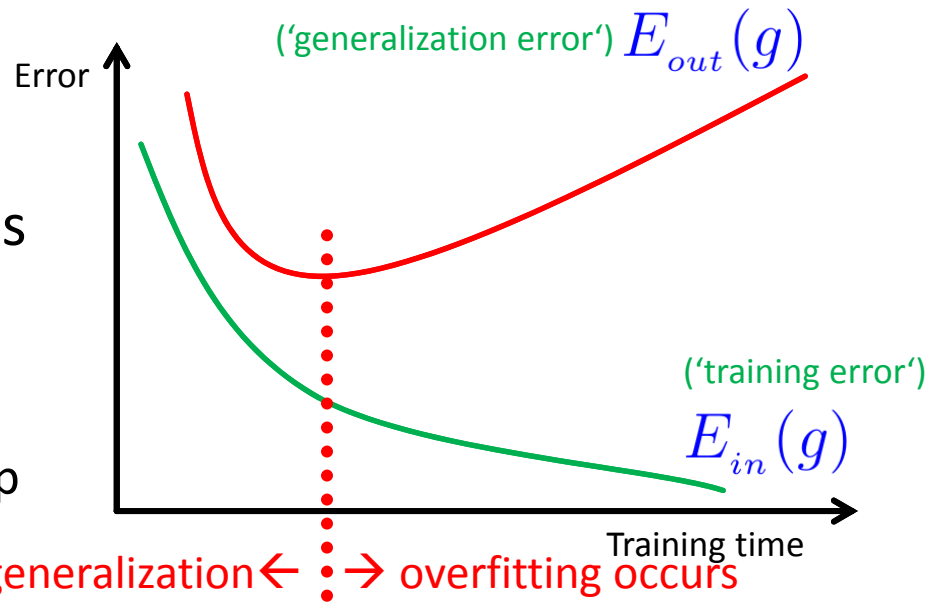


Problem of Overfitting – Clarifying Terms

- **A good model must have low training error (E_{in}) and low generalization error (E_{out})**
- **Model overfitting is if a model fits the data too well (E_{in}) with a poorer generalization error (E_{out}) than another model with a higher training error (E_{in})**

[2] Introduction to Data Mining

- Overfitting & Errors
  - $E_{in}(g)$ goes down
  - $E_{out}(g)$ goes up
- 'Bad generalization area' ends
  - Good to reduce $E_{in}(g)$
- 'Overfitting area' starts
  - Reducing $E_{in}(g)$ does not help
  - Reason 'fitting the noise'



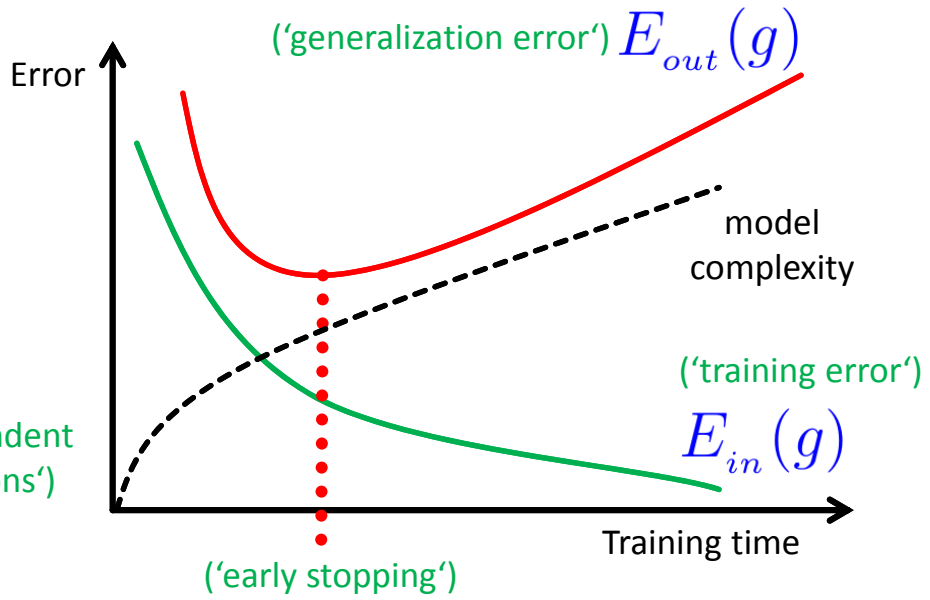
- **The two general approaches to prevent overfitting are (1) regularization and <u>(2) validation</u>**

(Decisions about the model are related to the problem of overfitting – need another method to 'select model well')

Problem of Overfitting – Model Relationships

- Review ‘overfitting situations‘
  - When comparing ‘various models‘ and related to ‘model complexity‘
  - Different models are used, e.g. 2nd and 4th order polynomial
  - Same model is used with e.g. two different instances (e.g. two neural networks but with different parameters)
- Intuitive solution
  - Detect when it happens
  - ‘Early stopping regularization term‘ to stop the training
  - Early stopping method (later)

(‘model complexity measure: the VC analysis was independent of a specific target function – bound for all target functions‘)



- **‘Early stopping‘ approach is part of the theory of regularization, but based on validation methods**

Problem of Overfitting – Impacts on Learning

- **The higher the degree of the polynomial (cf. model complexity), the more degrees of freedom are existing and thus the more capacity exists to overfit the training data**

- Understanding deterministic noise & target complexity
  - Increasing target complexity increases deterministic noise (at some level)
  - Increasing the number of data N decreases the deterministic noise
- Finite N case: $\mathcal{H}$ tries to fit the noise
  - Fitting the noise straightforward (e.g. Perceptron Learning Algorithm)
  - Stochastic (in data) and deterministic (simple model) noise will be part of it
- Two 'solution methods' for avoiding overfitting
  - Regularization: 'Putting the brakes in learning', e.g. early stopping (more theoretical, hence 'theory of regularization')
  - Validation: 'Checking the bottom line', e.g. other hints for out-of-sample (more practical, methods on data that provides 'hints')

Regularization – Two Approaches

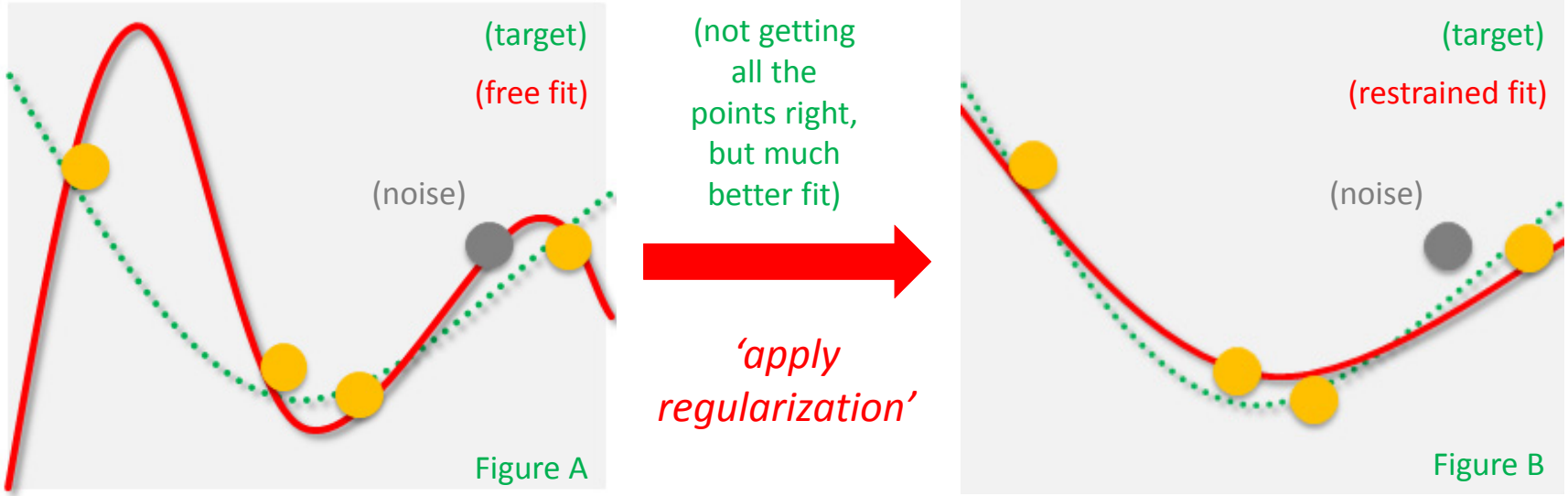
- Used in almost every machine learning situation
  - Two approaches to regularization: mathematics & heuristics
- Mathematics (from ill-posed problems in function approximation)
  - Goal: approximate a function
  - 'Ill-posed Problem': there are many functions that fit this function
  - Approach: Impose 'smoothness constraints' to solve the problem
  - Learning practice: assumptions in mathematical approach not realistic
  - Solution: use mathematical approach intuitively to create 'rules of thumb'
- Heuristics
  - Handy-capping the minimization of the in-sample-error
  - Idea of 'putting the brakes' – partly based on mathematics $E_{in}(g)$
  - Not a random process and oriented towards preventing overfitting

- **Regularization creates simpler models & thus in-line with 'Occams Razor' (simple model better)**

Regularization – Key Principle ‘Do not take data too serious‘

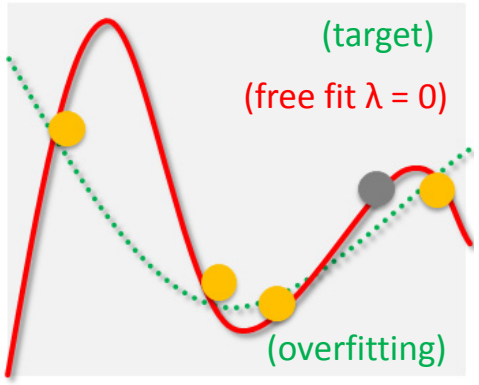
- Initial setup → Rather memorizing (not learning to generalize)
  - Considered as ‘free fit‘ – ‘fit as far as the model can do‘
  - E.g. use 4th order polynomial model and fit the 5 data points (cf. Figure A)
- ‘Putting the brakes‘ regularization to avoid overfitting
  - Apply a ‘restrained fit‘ – ‘preventing to fit the data perfectly‘
  - E.g. use 4th order polynomial model but use ‘minimal brakes‘ (cf. Figure B)

(equivalent meaning of explicitly forbidding some of the hypothesis to be considered in learning)

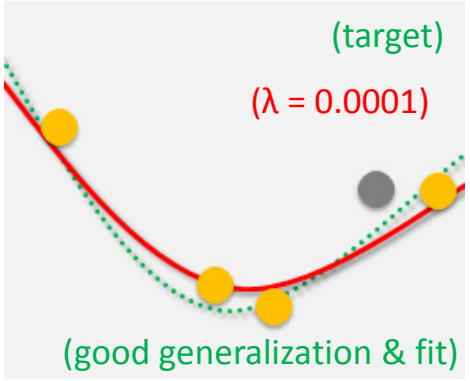


Regularization – Regularization Example & Impact

- Example: Minimize $E_{in}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$ (penalizing models with extreme parameter values)

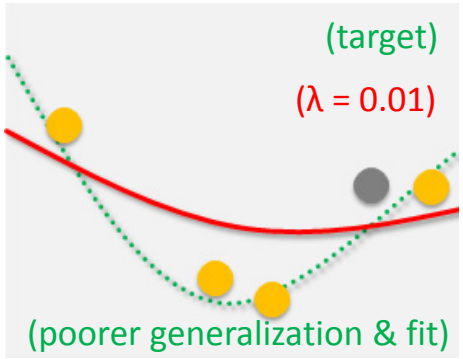


'apply little regularization'

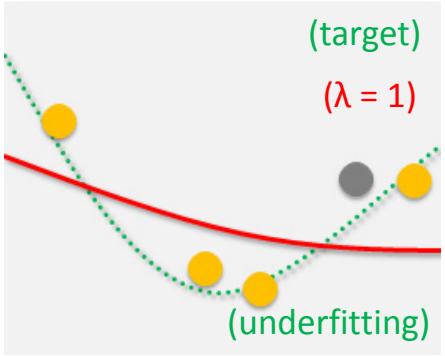


(regularization coefficient λ controls tradeoff: how much it is important to fit the data well vs. simple hypothesis)

'apply more regularization'



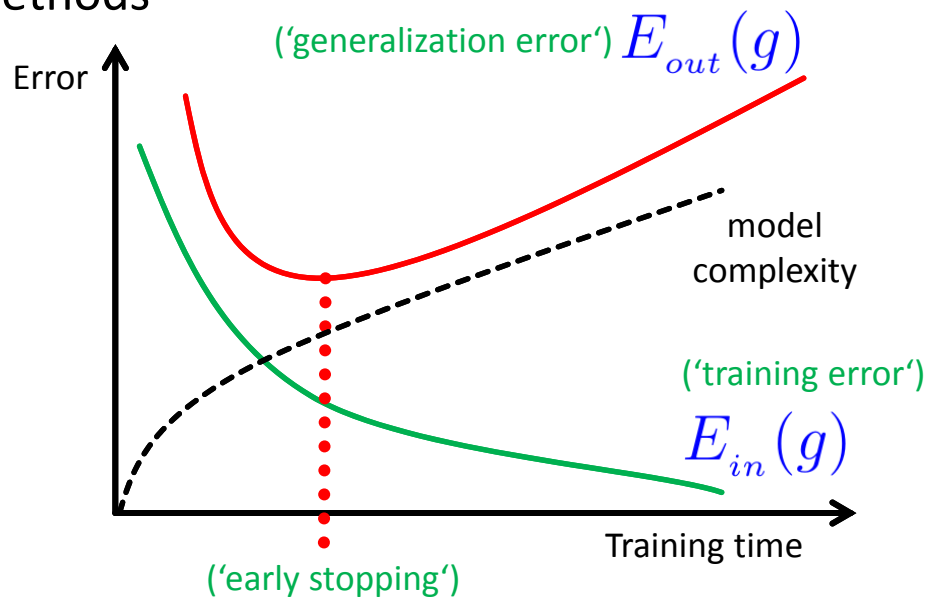
'apply too much regularization'



- **Choice of λ is extremely critical and can lead with high values to underfitting (as bad as overfitting)**
- **Choice of type of regularizers is heuristic, but choice of λ will be extremely principled (validation)**

Regularizer Methods – Early Stopping & Big Data

- Idea: regularization through the optimizer
  - Apply early stopping algorithm as part of the 'training time'
  - Based on practical validation methods
  - Point in time to stop will be choosen by validation



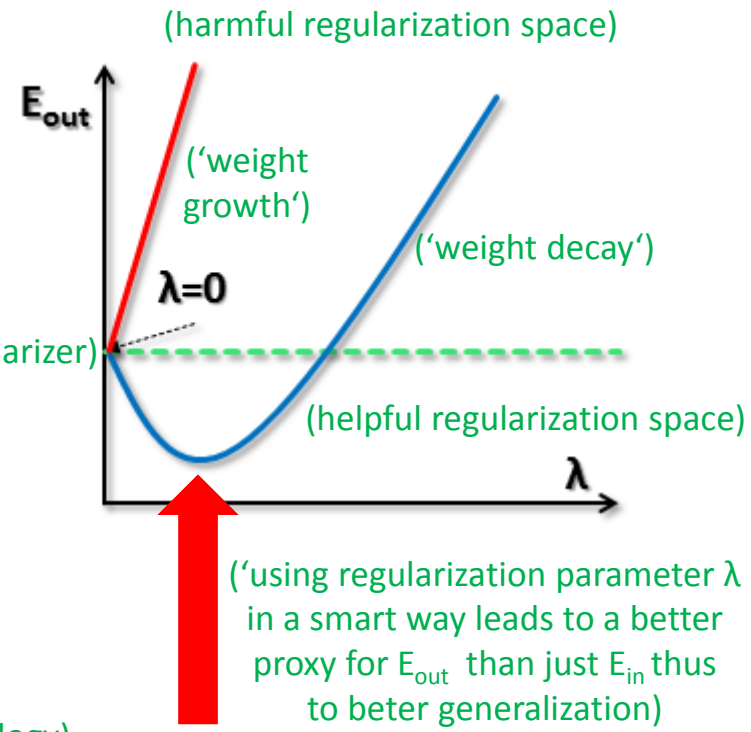
- **Big data has a bigger dataset N and more likely also 'more stochastic noise'**
- **Preventing overfitting with big data tends to require stricter regularization**
- **Early stopping offers a pragmatic way for big data to prevent overfitting**
- **Given large quantities of dataset N, the training time is massively reduced too**

Regularization Methods – Choosing Guidelines

- **Choosing a regularizer in learning from data is a must have & leads to better generalization**
- **Choosing not a regularizer will definitely lead to overfitting the data in practical situations**

- Selected ‘rules of thumb‘
  - Choosing constraints with ‘weight growth‘ gives terrible results
  - Choosing constraints towards ‘smoother hypothesis‘ gives good results
- Reasoning noise is not smooth
  - Stochastic noise → ‘high frequency‘
  - Determinisic noise → ‘non-smooth‘
  - Smaller weights correspond to smoother hypothesis

(‘hectic dog & calm man have a walk‘ analogy)



- **The rule of thumb in regularization is to constrain the learning with smoother/simpler hypothesis**
- **Regularization forbids some hypothesis & thus reduces the VC dimension: improving generalization**

Term Support Vector Machines – Revisited

- **Support Vector Machines (SVMs) are a classification technique developed ~1990**
- **SVMs perform well in many settings & are considered as one of the best ‘out of the box classifiers‘**

[3] An Introduction to Statistical Learning

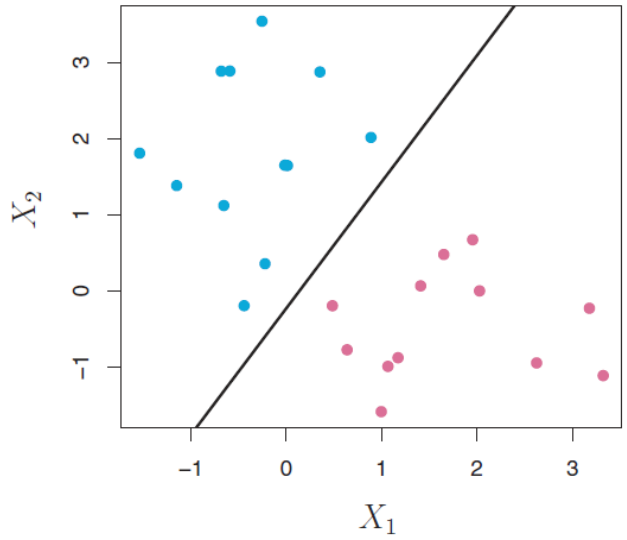
- Term detailed refinement into ‘three separate techniques’
  - Practice: applications mostly use the SVMs with kernel methods
- ‘Maximal margin classifier‘
  - A simple and intuitive classifier with a ‘best‘ linear class boundary
  - Requires that data is ‘linearly separable‘
- ‘Support Vector Classifier‘
  - Extension to the maximal margin classifier for non-linearly seperable data
  - Applied to a broader range of cases, idea of ‘allowing some error‘
- ‘Support Vector Machines‘ → Using Non-Linear Kernel Methods
  - Extension of the support vector classifier
  - Enables non-linear class boundaries & via kernels;

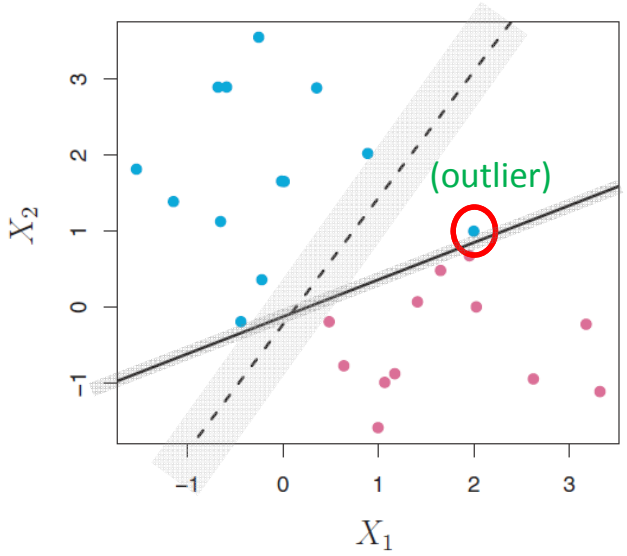
Support Vector Classifiers – Motivation

- **Support Vector Classifiers develop hyperplanes with soft-margins to achieve better performance**
- **Support Vector Classifiers aim to avoid being sensitive to individual observations (e.g. outliers)**

[3] An Introduction to Statistical Learning

- Approach
  - Generalization of the ‘maximal margin classifier‘ (get most & but not all training data correcly classified)
  - Include non-seperable cases with a soft-margin (almost instead of exact)
  - Being more robust w.r.t. extreme values (e.g. outliers) allowing small errors





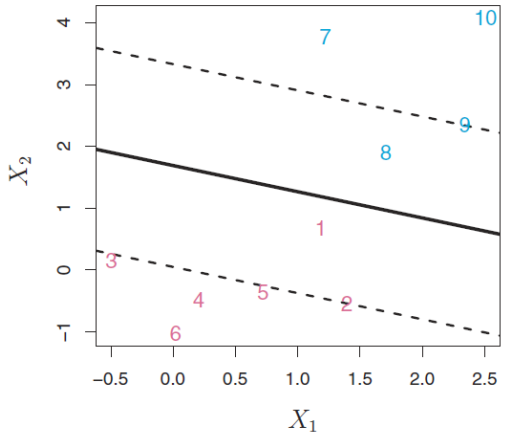
(significant reduction of the maximal margin)

(overfitting, cf Lecture 10: maximal margin classifier & hyperplane is very sensitive to a change of a single data point)

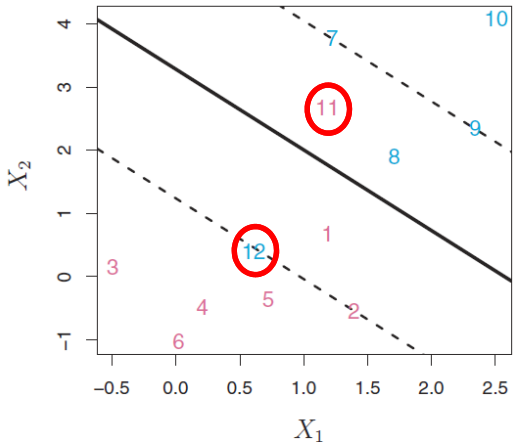
Support Vector Classifiers – Modified Technique Required

- Previous classification technique reviewed
  - Seeking the largest possible margin, so that every data point is…
  - … on the correct side of the hyperplane
  - … on the correct side of the margin
- Modified classification technique
  - Enable ‘violations of the hard-margin‘ to achieve ‘soft-margin classifier‘
  - Allow violations means: allow some data points to be…
  - … on the incorrect side of the margin
  - … even on the incorrect side of the hyperplane (which leads to a misclassification of that point)

(incorrect side of margin)



(incorrect side of hyperplane)



[3] An Introduction to Statistical Learning

(SVs refined: data points that lie directly on the margin or on the wrong side of the margin for their class are the SVs → affect support vector classifer)

Support Vector Classifier – Optimization Problem Refined

- Optimization Problem
  - Still maximize margin M
  - Refining constraints to include violoation of margins
  - Adding slack variables $\epsilon_1, \ldots, \epsilon_n$

$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

(allow datapoints to be on the wrong side of the margin or hyperplane)

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$ (slightly violate the Margin)

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C$$

(C is used here to bound the error)

(C is a nonnegative tuning parameter useful for regularization, cf. Lecture 10, picked by cross-validation method)

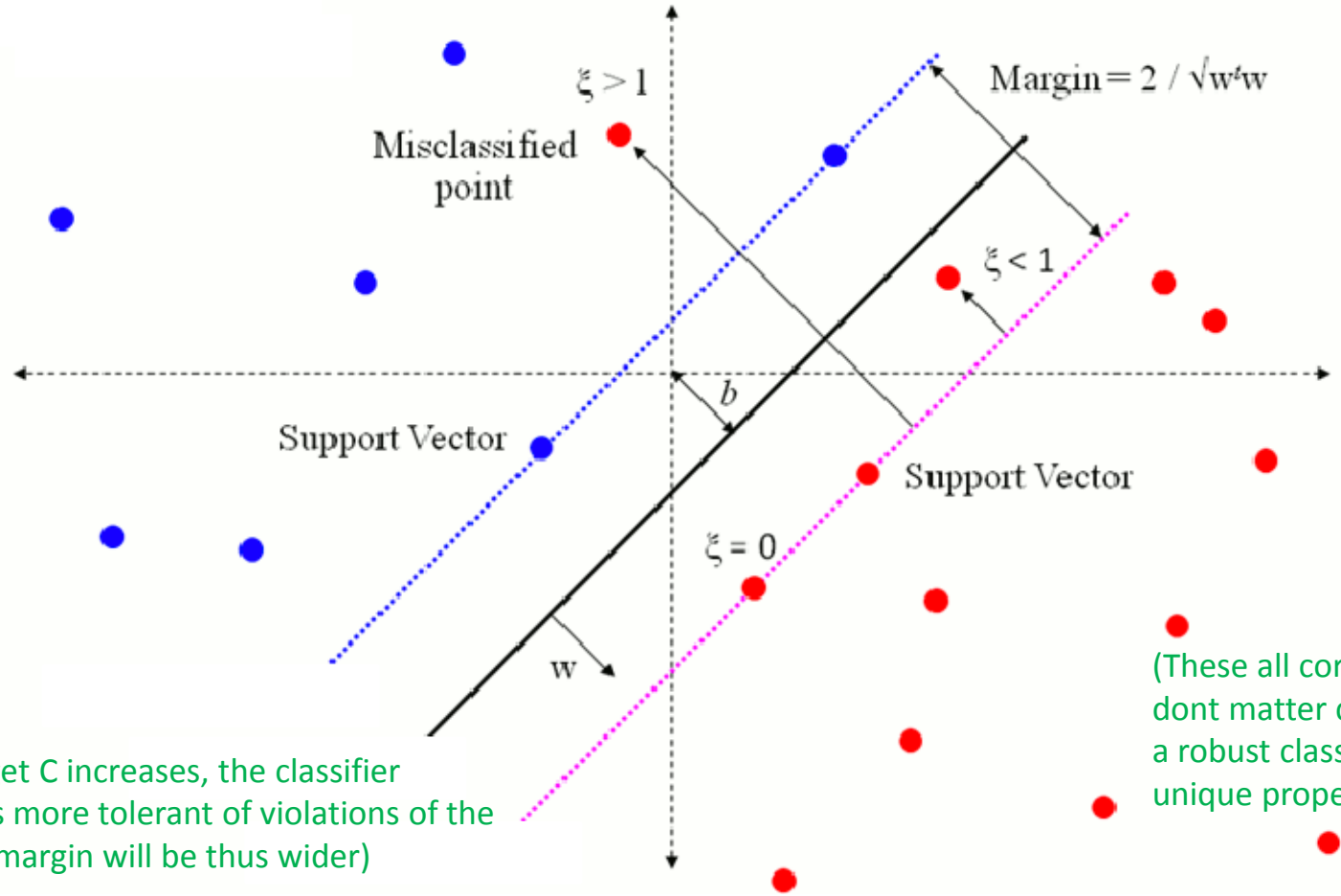
- C Parameter & slacks
  - C bounds the sum of the slack variables $\epsilon_1, \ldots, \epsilon_n$

[3] An Introduction to Statistical Learning

- **C determines the number & severity of violations that will be tolerated to margin and hyperplane**
- **Interpret C as budget (or costs) for the amount that the margin can be violated by n data points**

Support Vector Classifier – Understanding the Slack

- Allowing some errors or violations of the margin



(As budget C increases, the classifier becomes more tolerant of violations of the margin, margin will be thus wider)

(These all correct samples dont matter contributing to a robust classifier and unique property of SVCs)

Support Vector Classifier – Impact of Regularization with C

- Approach: Maximizing the margin
  - Equivalent to minimize objective function

$$\min_{w} \left\{ \frac{1}{2} \| \mathbf{w} \|^2 \right\}$$

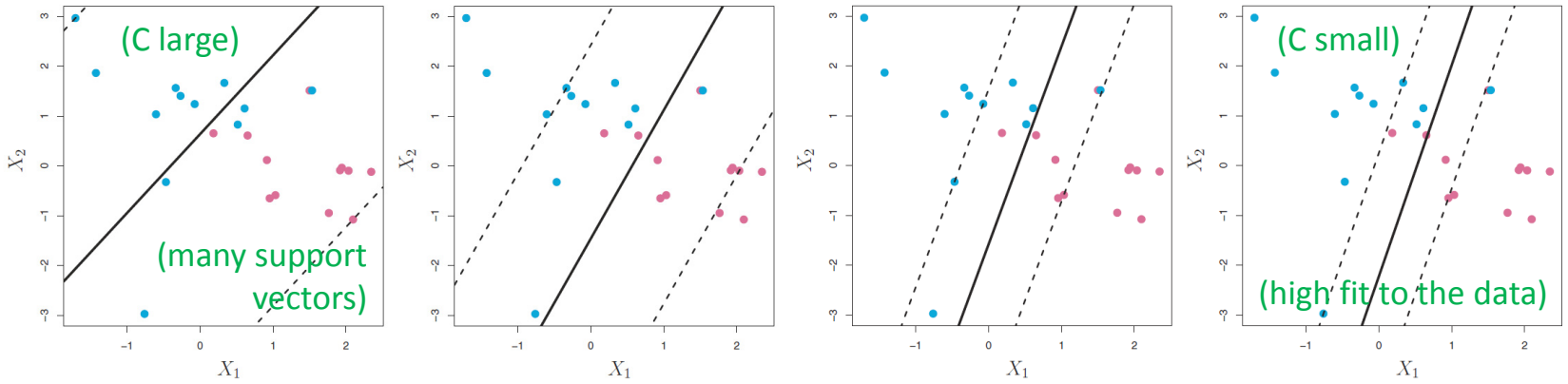
(maximal margin classifier)

- Support Vector Classifier
  - Same approach for solving
  - Adding slacks variables
  - C parameter that enables regularization (i.e. size of margin)

$$\min_{w,\xi_i} \left\{ \frac{1}{2} \| \mathbf{w} \|^2 + C \sum_i \xi_i \right\}$$

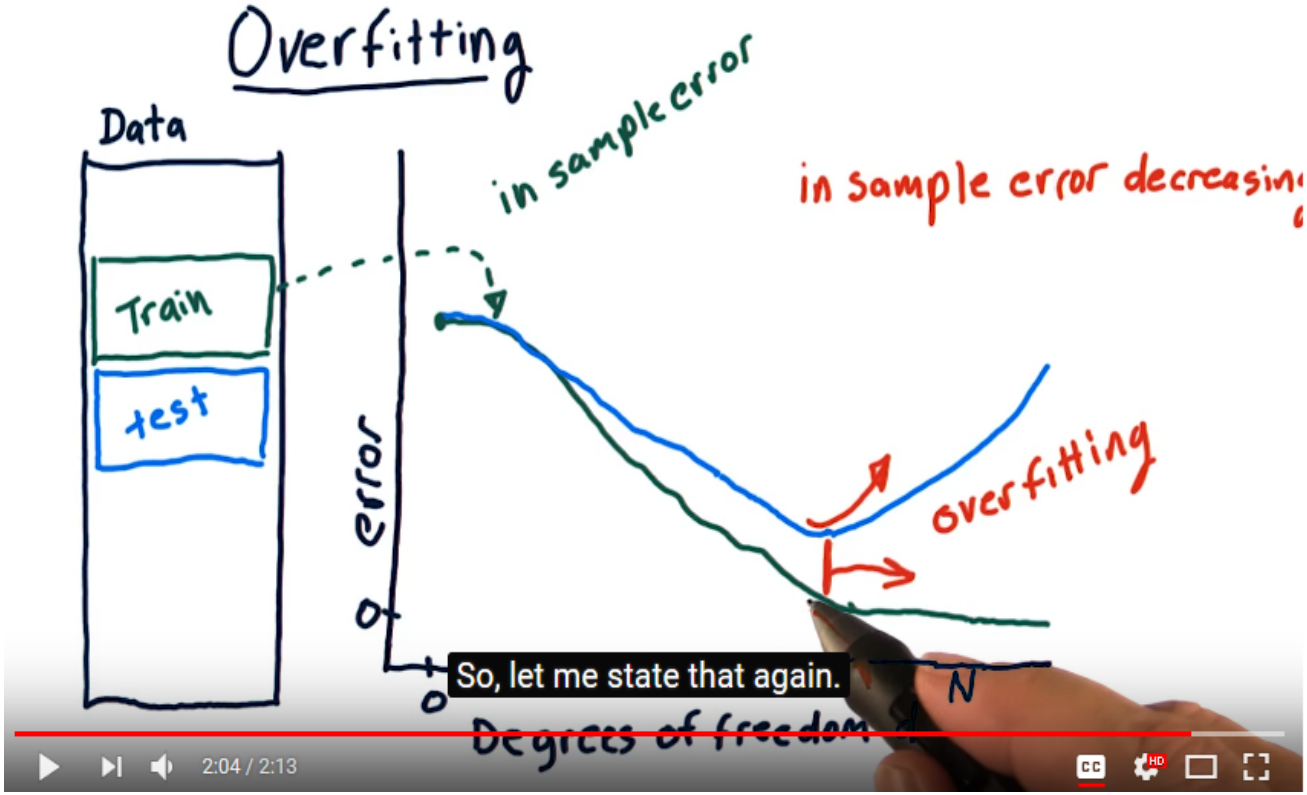
(C is used here to multiply the sum of errors for increased severity, because # errors should be kept small, not a budget here)

[3] An Introduction to Statistical Learning



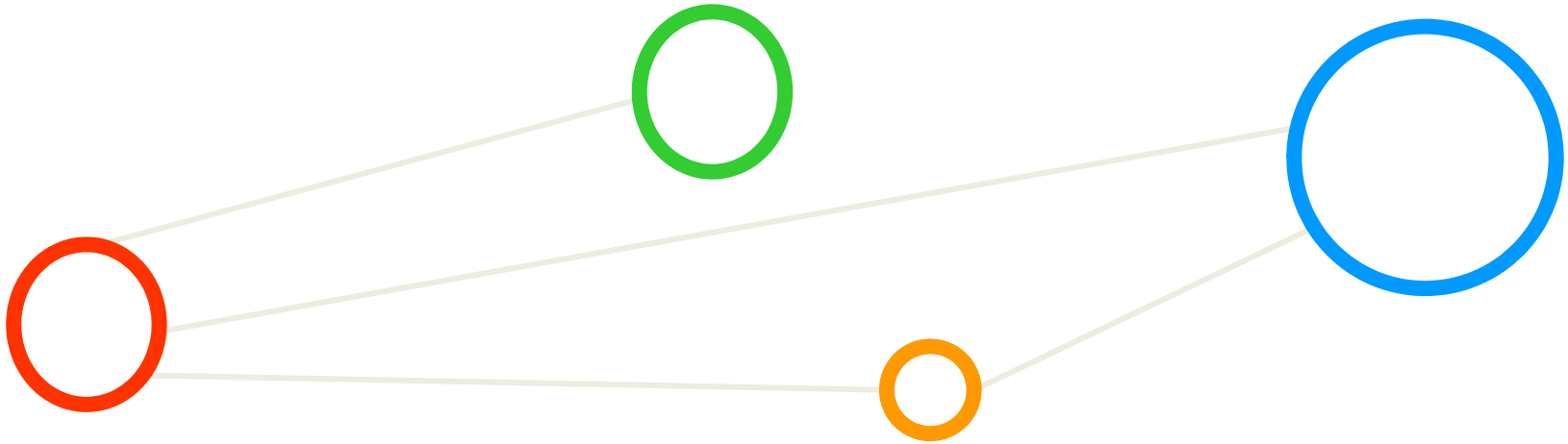
(fewer support Vectors: just #8)

[Video] Overfitting Challenge



[6] YouTube Video, 'Overfitting'

Lecture Bibliography



Lecture Bibliography (1)

- [1] G. Cavallaro, M. Riedel, M. Richerzhagen, J. A. Benediktsson and A. Plaza, "On Understanding Big Data Impacts in Remotely Sensed Image Classification Using Support Vector Machine Methods," *in the IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 10, pp. 4634-4646, Oct. 2015.
- [2] Introduction to Data Mining, Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Addison Wesley, ISBN 0321321367, English, ~769 pages, 2005
- [3] An Introduction to Statistical Learning with Applications in R, Online: http://www-bcf.usc.edu/~gareth/ISL/index.html
- [4] Rome Dataset, B2SHARE, Online: http://hdl.handle.net/11304/4615928c-e1a5-11e3-8cd7-14feb57d12b9
- [5] YouTube Video, 'Machine Learning : Model Selection & Cross Validation', Online: http://www.youtube.com/watch?v=hihuMBCuSlU
- [6] YouTube Video, 'Overfitting', Online: https://www.youtube.com/watch?v=mfzHchd5La8
- [7] Y.S. Abu-Mostafa, M. Magdon-Ismail, Hsuan-Tien Lin, 'Learning from Data – A short course', AML Book, ISBN 978-1-60049-006-4

Slides Available at http://www.morrisriedel.de/talks