Data-driven Intelligent Systems

Lecture 11

Theory of Learning, Evaluation

KNOWLEDGE
TECHNOLOGY

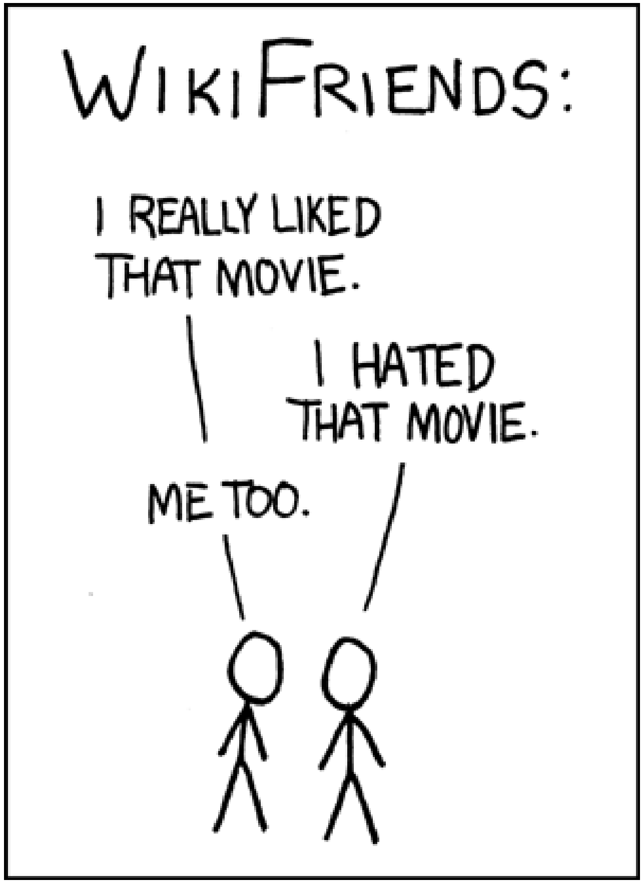
http://www.informatik.uni-hamburg.de/WTM/

Theory of Learning from Data

- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

Machine Learning & Human Learning

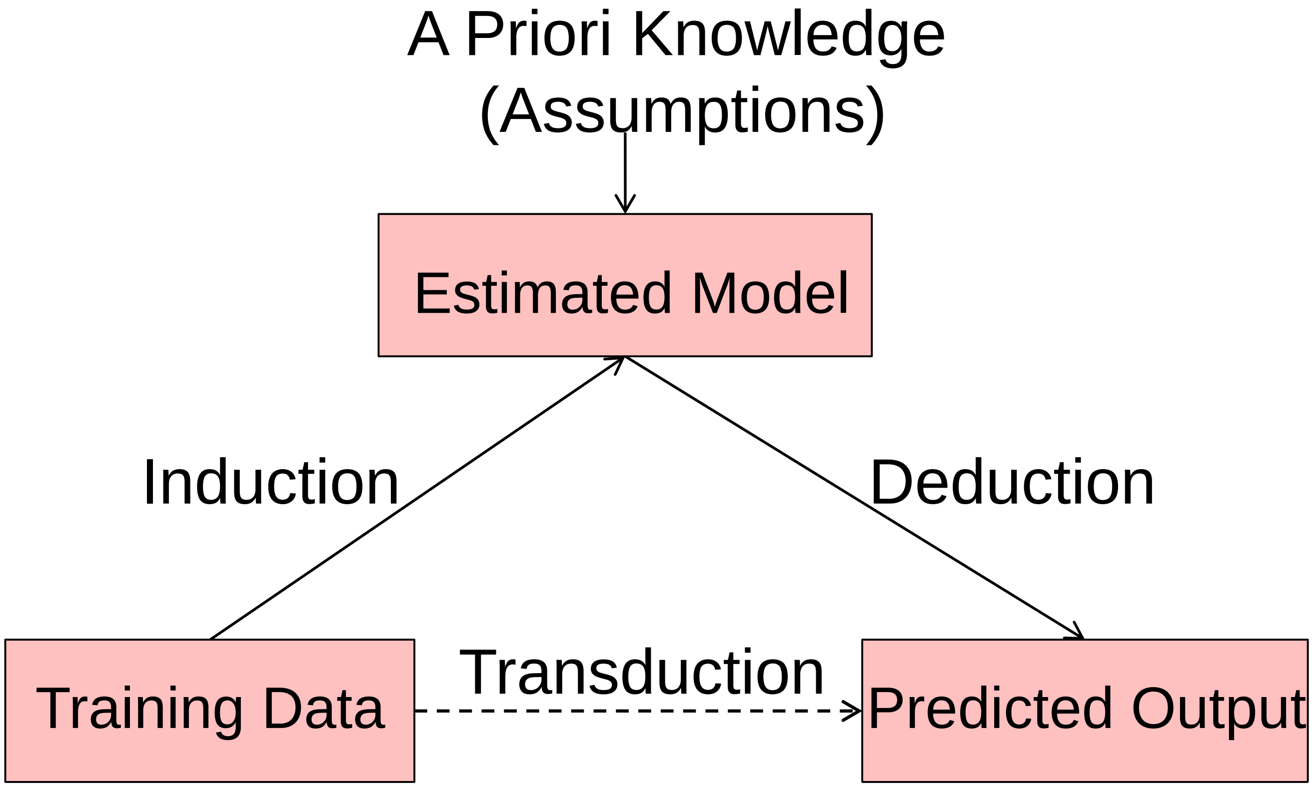
- Supervised, unsupervised, semi-supervised, self-supervised, reinforcement learning
- Learning from examples
- Case-based learning
- Learning by analogy
- Learning by doing
- …



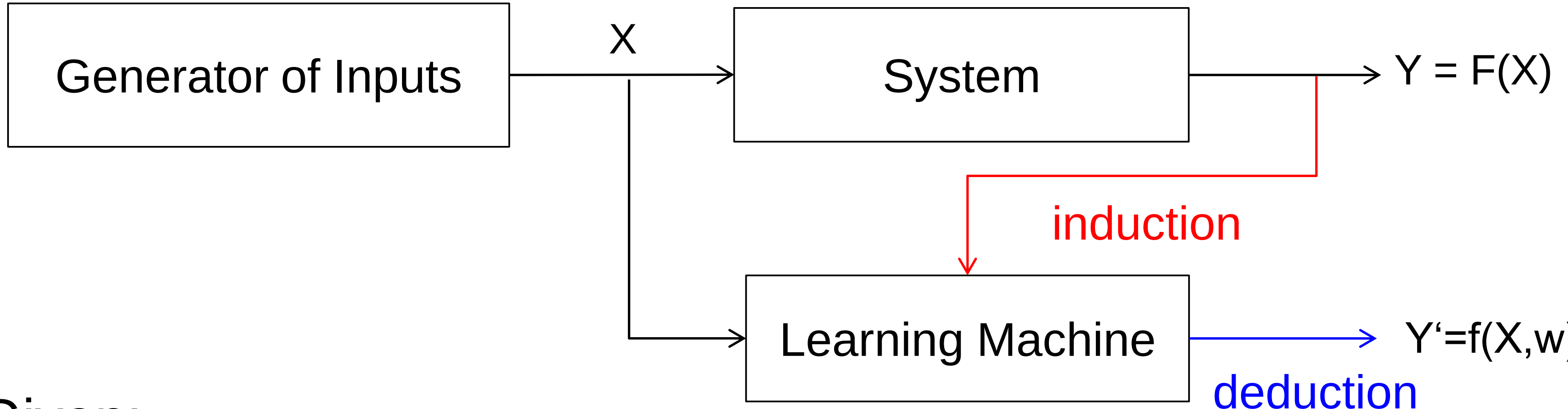
Machine Learning Issues

- Static vs. dynamic data
- Centralized vs. distributed data
- Batch vs. incremental (on-line) learning
- Active/adaptive learning
- Life-long learning
- …

Types of Inference: Induction, Deduction, Transduction



A Learning Scenario



Given:

- observed samples {(X, Y)}

How to select f(X, w):

- Approximating function f?
  - Hyperparameters?
- Parameters: w?

← A priori knowledge required!

Example:

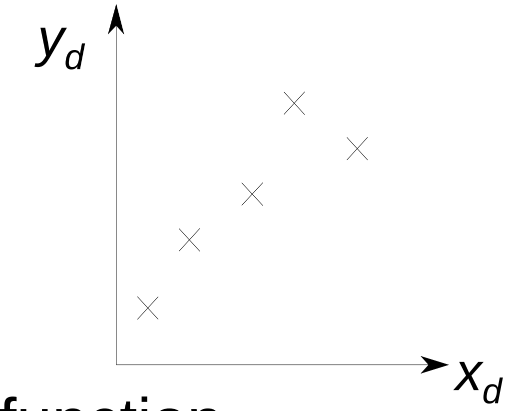
f: linear in parameters:

$$y = w_1 x^n + w_2 x^{n-1} + \cdots + w_0$$

nonlinear in parameters:

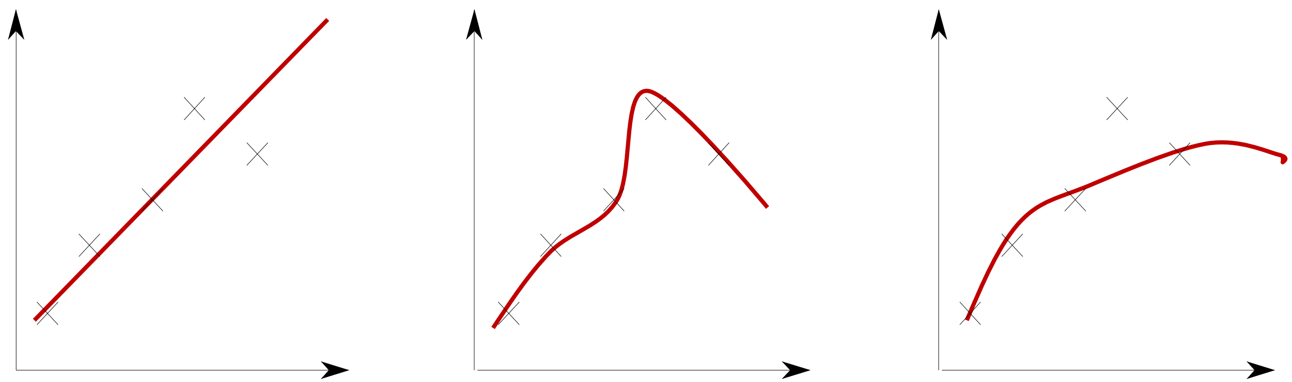
$$y = e^{-wx}$$

Hypotheses for a Given Data Set



- Given: samples (x_d, y_d)
- Unknown: true function $y=F(x)$
- Wanted: approximation $\underbrace{h(x)}_{\text{hypothesis}}$ of the true function

Polynomial (linear, quadratic, etc.) or exponential model?



How to Learn with a Learning Machine? (1)

- Learning objective
  - *Inductive principle* – a general prescription for learning
  - Tells us ***what*** we wish to achieve with the data

  → define a Risk function

  → choose a model (approximating function) of suitable complexity

- Learning method
  - Tell us ***how*** to obtain an optimal estimate
  - I.e. a constructive implementation of an inductive principle

  → find good model parameters

How to Learn with a Learning Machine? (2)

- **Loss function** $L(y_d, f(x_d, w))$: (also: **Error function**)
  - measure of difference between y_d and $f(x_d, w)$ for each sample d
    - y_d – the output produced by the system,
    - X_d – a tuple of inputs,
    - $f(X_d,w)$ – the output produced by the learning machine for a selected approximating function f,
    - w – the set of parameters in the approximating function.
- **Risk functional** R(w):
  - measure of accuracy of the learning machine:

$$R(w) = \frac{1}{\#d} \cdot \Sigma_d \; L(y_d, f(x_d, w))$$

Analogue terms: **Cost**, Score, Profit, Fitness, Utility, Reward, **Objective** function

How to Learn with a Learning Machine? (3)

- Examples of loss function $L(y, f(x, w))$:
  - **Classification error**:

$$L(y, f(x, w)) = \begin{cases} 0 & \text{if } \quad y = f(x, w) \\ 1 & \text{if } \quad y \neq f(x, w) \end{cases}$$

  - **Squared error** (a measure for regression):

$$L(y, f(x, w)) = (y - f(x, w))^2$$

Theory of Learning from Data

- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

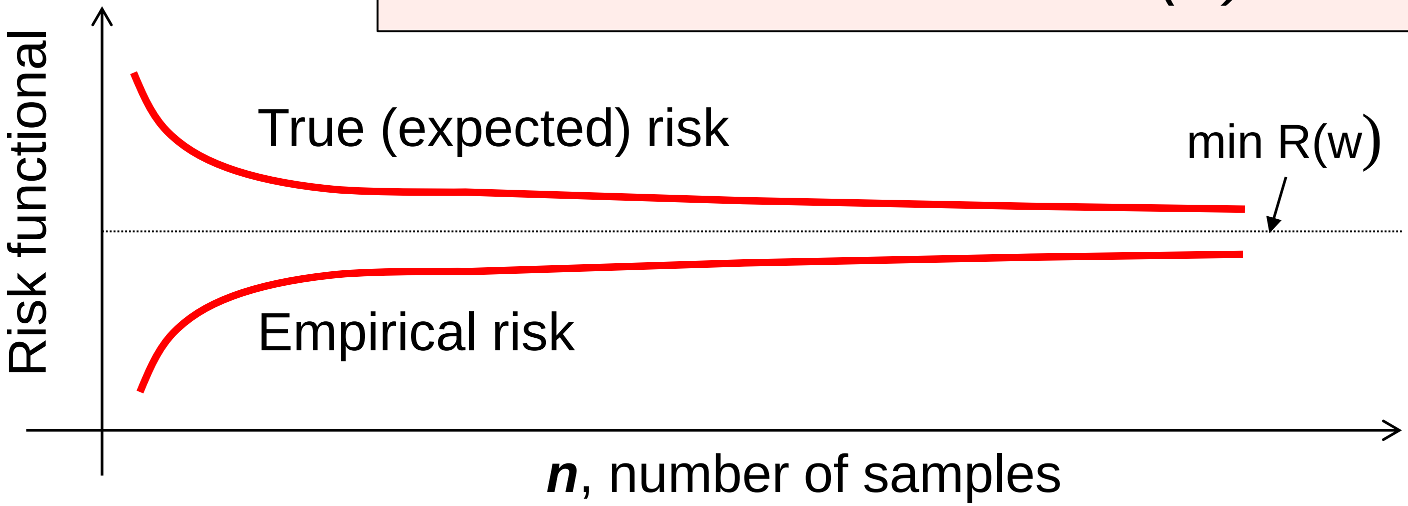
Statistical Learning Theory (1)

- SLT – formalizes many learning procedures developed in AI, ANN, statistics, Data Mining, Pattern Recognition
- SLT considers learning with small sets of samples
  - → Exact distribution of data *p(x, y)* is unknown
  - → When does overfitting occur?
  - → Approximate true risk *R(w)* with an empirical risk
- ***Empirical Risk Minimization (ERM)*** – the basic inductive principle:
  - Find the optimal estimate = minimum of risk function *R(w)* based only on the available data
  - Implementation of ERM depends on selected *L* and *f(x, w)*
- SLT = VC theory (**V**apnik **C**hervonenkis)

Statistical Learning Theory (2)

- Asymptotically consistent estimator:

  when $n \rightarrow \infty$ then true $R(w) \approx$ empirical risk



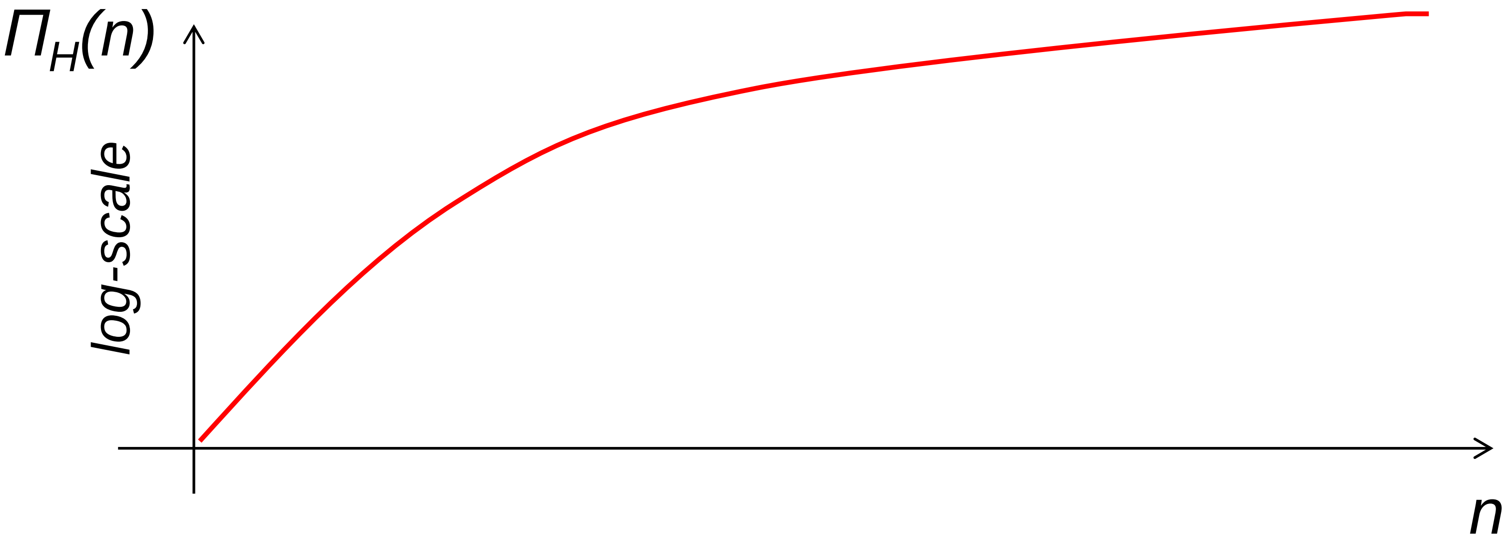
Assume:
- given data distribution
- fixed number of model parameters
- model fully trained for each ***n***

- For $n \rightarrow \infty$
  - true model parameter values will be estimated
  - model will generalize to "unseen" data
- Asymptotic consistency should hold for ALL classes of approximating functions

Statistical Learning Theory (3)

- To ensure ***asymptotic consistency***, approximating functions should be like a ***growth function***
- As the number of samples grows, the approximating functions should start to ***generalize***
- Generalization means
  - failure to model noise
  - failure to model overly complex data
- The set of hypothesis that the approximating function can make over the data should be limited

Growth Function



- Hypothesis set H = all the functions a learner can approximate
- A growth function is defined as

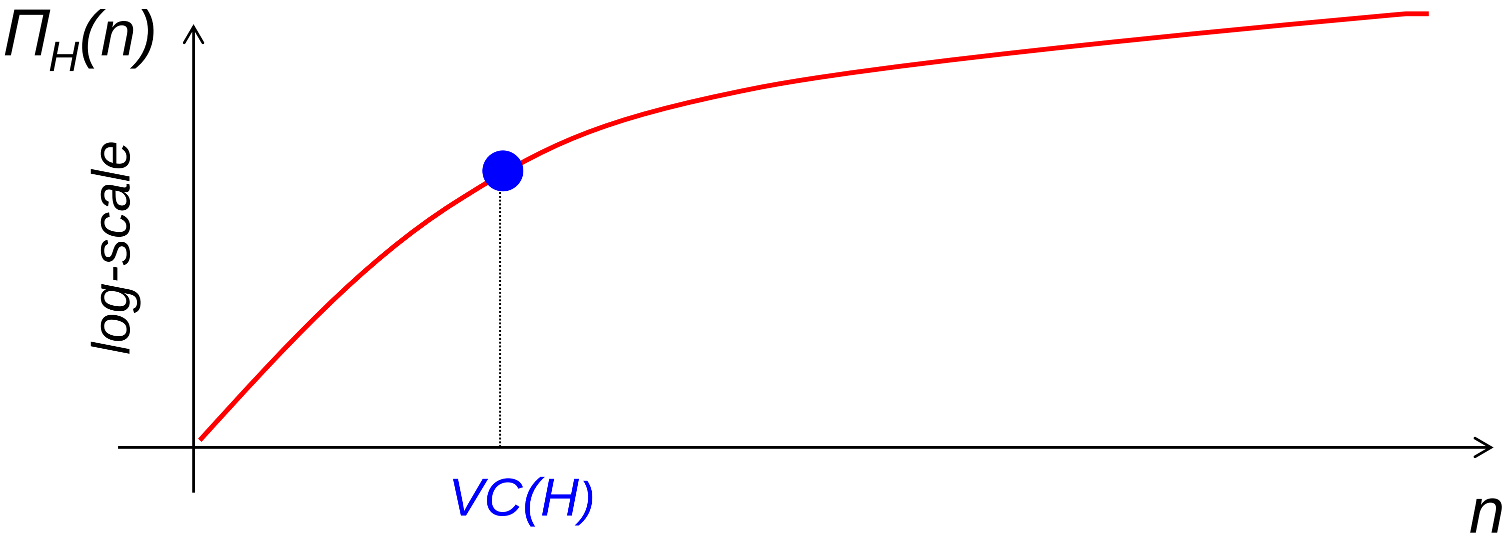
$$\Pi_H(n) = \max |H(S)|$$

over all input sets S of size n

i.e. the maximum number of ways n points can be classified by H

- E.g. binary classification: $\Pi_H(n) \leq 2^n$

Vapnik Chervonenkis (VC) Dimension



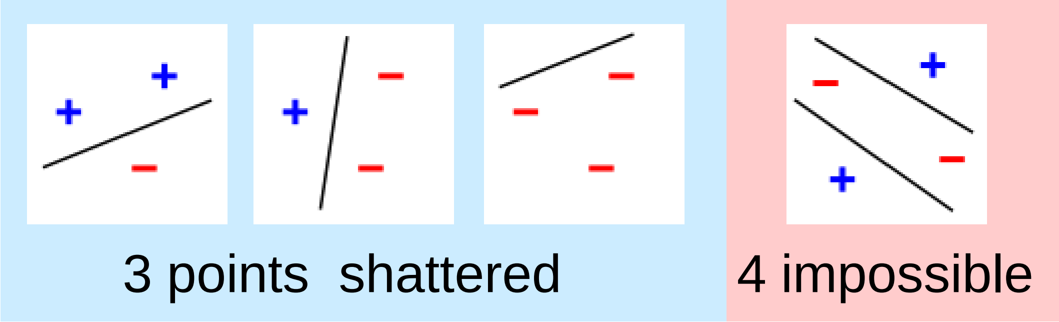
VC dimension: Point ***n = VC*** where growth starts to slow down

- The *VC* dimension of *H* is the cardinality of the largest set *S* that can be fully represented by *H* (i.e. learned)
- *VC(H)* is typically finite in good learners
- A “saturating” growth function ensures asymptotic consistency

VC Dimension, Examples

- Linear classifier in 2D:

  VC(H) = 3



- Linear classifiers for *d* features plus a constant term *b*:

  VC(H) = *d*+1 (perceptron)

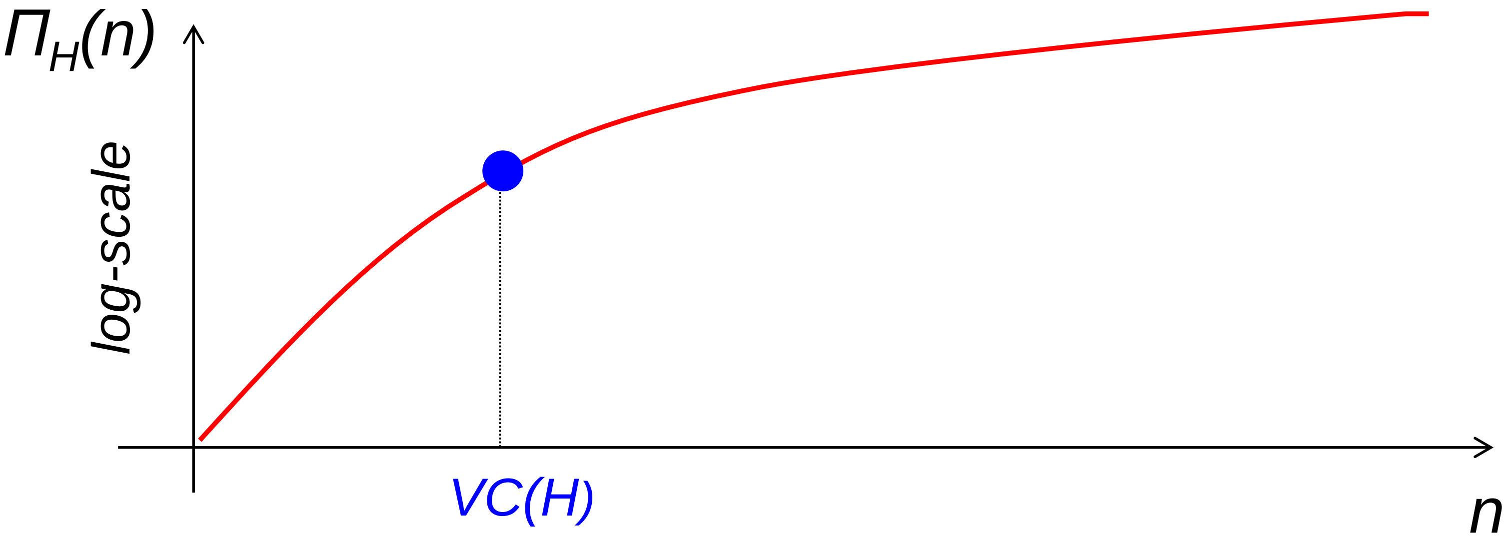
- Neural networks:

  VC(H) ≈ #parameters

- Decision tree of rank *r* that defines Boolean functions on *n* boolean variables:

$$\text{VC(H)} = \sum_{i=0}^{r} \binom{n}{i}$$

VC Dimension

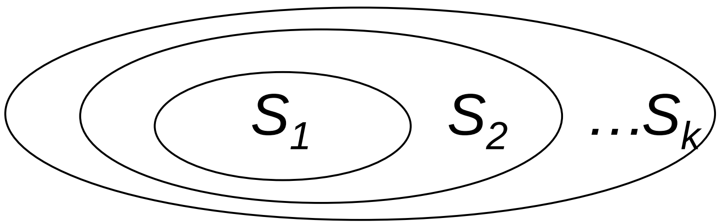


VC dimension*:** Point ***n = VC where growth starts to slow down

- **ERM** applicable for large *n* ($n/VC > 20$)
- Possible overfitting for small *n* ($n/VC < 20$)
  → need to constrain the structure of the learner → **SRM**

Structural Risk Minimization (1)

- SRM requires a priori specification of a structure for sets of approximating functions $S_1, S_2, \ldots S_k$.



$$VC(S_1) < VC(S_2) < \ldots < VC(S_k)$$

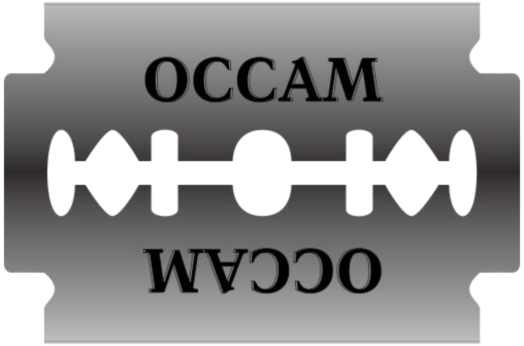
- ***SRM approach*** towards optimal model:
  - Calculate or estimate *VC*-dimension for any element S_k
  - Minimize empirical risk $R(w)$ for each S_k
- The optimal solution is a tradeoff:
  - High complexity (large *VC*) $\rightarrow$ small empirical risk
  - Low complexity (small *VC*) $\rightarrow$ empirical risk ~ true risk

good genera-lization

Structural Risk Minimization (2)

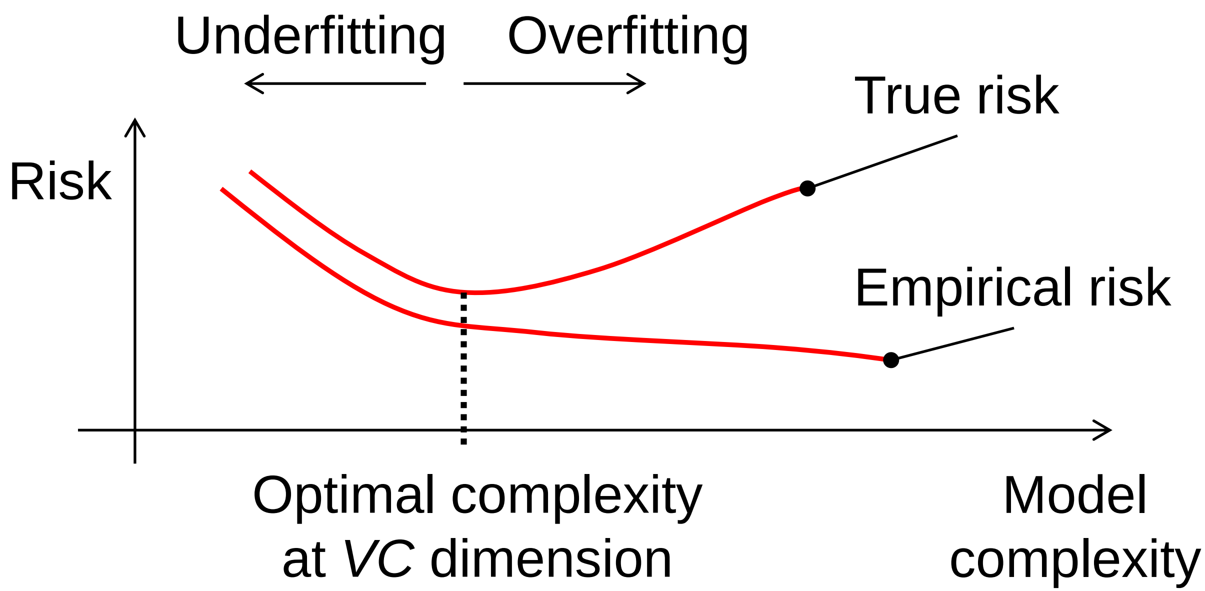
- SRM – a trade off between ***complexity*** (of approximating functions) and ***quality*** (of results)

> ”As simple as possible,
> but with enough quality.“
>
> Occam‘s razor principle



- Optimal model estimation:
  - Select an element of the structure with optimal complexity
  - Define the model based on selected approximating functions
  - Penalize complex models by ***regularisation***

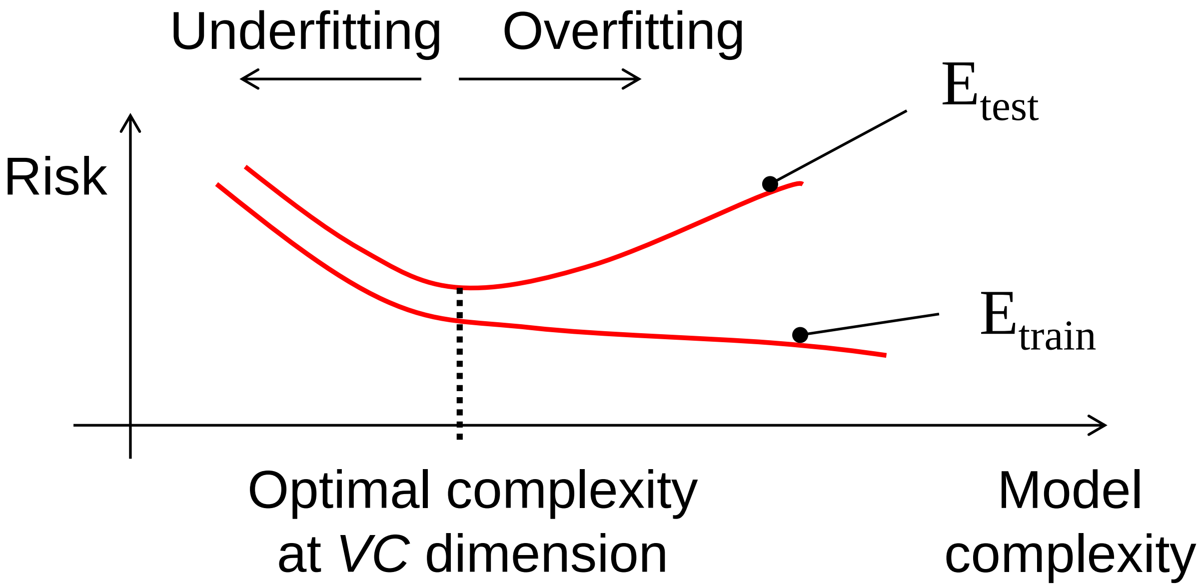
SRM Optimization Strategy



With increasing complexity of approximating functions true & empirical risk *R(w)* decrease until the value – *VC* dimension; thereafter they diverge.

- Optimization:
  - Stochastic approximation (or gradient descent)
  - Iterative methods
  - Greedy optimization (following locally optimal choice)

Complexity and Generalization



Test error estimates true risk

Training error measures empirical risk

- Complexity = degrees of freedom in the model
  - **E.g.**: number of variables
  - Effective model complexity may rise over the course of training (this justifies early stopping)

Bias-Variance Tradeoff

- Model ***bias***:

  may result from SRM

  - Model outputs are often biased – models can learn certain aspects of the data, but have limitations elsewhere
    - Underfitting is a form of bias
  - Model bias unwanted, since output shall depend on the data
  - But: a smart bias may enable certain model performance!
- Model ***variance***:

  may result from ERM

  - Models' outputs often have large variance under:
    - small variations in the data, e.g. different sampling, or
    - with different initial random values of the model parameters
  - Unwanted variance often observed in powerful models, which are unconstrained by model bias
    - Overfitting models have this behaviour

https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff

Theory of Learning from Data

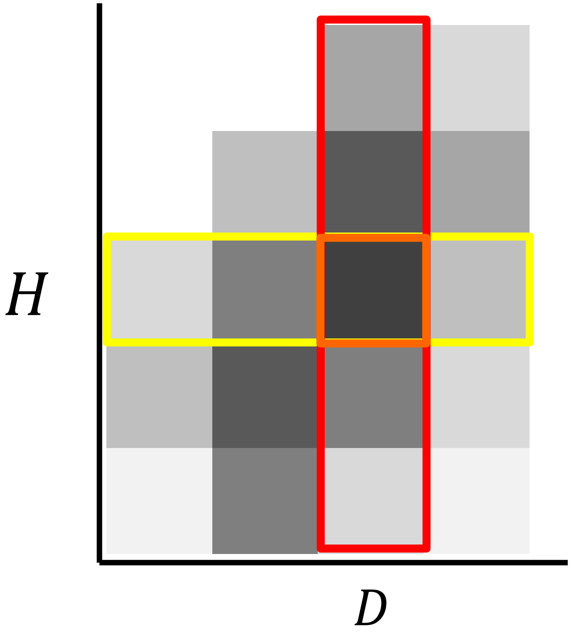
- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

Bayes

- Probability distribution of two random variables:

$$\boxed{P(D,H)} = P(D \mid H)\cdot \boxed{P(H)}$$

$$= P(H \mid D)\cdot \boxed{P(D)}$$



- Rearrange terms:

$$\boxed{P(H \mid D) = \frac{P(D \mid H)\cdot P(H)}{P(D)}}$$

Relation of ERM/SRM to Bayesian View

Bayes Theorem: $$P(H \mid D) = \frac{P(D \mid H) \cdot P(H)}{P(D)}$$

- *P(D|H)*: **Likelihood** that model *H* generates the data *D.*
  Find maximum likelihood model ~ ERM
  ~ try to model the data best, at any price.
- *P(H)*: **Prior** probability of model *H*; penalizes models of complex structure; based on *a priori* knowledge.
- *P(D)*: Evidence; just a normalizing factor.
- *P(H|D)*: **Posterior** probability for *H* after having seen the data.
  Find maximum posterior model
  Tradeoff: well performing & simple.
  ~ SRM

Relation Likelihood vs. Empirical Risk

- Likelihood for the model to generate the data:

maximise

$$P(D\,|\,H) \sim \prod_{data\ d} e^{-(y_d - f(x_d, w))^2}$$

Gaussian prob. of data deviating from model

- Take *–ln(.)* on both sides of the equation:
- Formulation as cost function:

$$-\ln(P(D\,|\,H)) \sim \sum_{data\ d} (y_d - f(x_d, w))^2$$

minimise

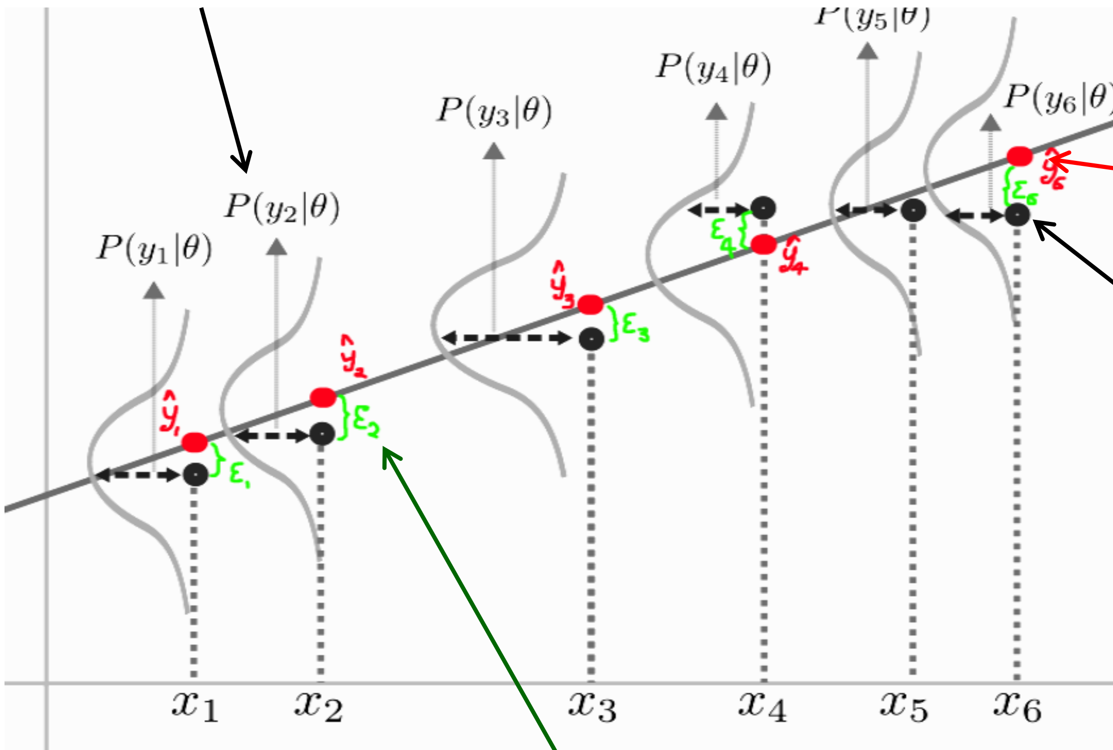
Square error

(empirical) Risk

Relation Likelihood vs. Empirical Risk

Likelihood probability

(of data being produced by the model)



model's predictions

data

Assumption:
Gaussian-deviated
data around model

error
(between data and prediction)

Probabilities vs. Cost Functions

- Bayes probabilistic formulation:

maximise $\longrightarrow$

$$\underbrace{P(H \mid D)}_{\text{Posterior}} \sim \underbrace{P(D \mid H)}_{\text{Likelihood}} \cdot \underbrace{P(H)}_{\text{Prior}}$$

- Take *–ln(.)* on both sides of the equation:

minimise $\rightarrow$

$$\underbrace{-\ln(P(H \mid D))}_{\text{Structural Risk}} \sim \underbrace{-\ln(P(D \mid H))}_{\text{Empirical Risk}} \underbrace{-\ln(P(H))}_{\text{Penalty on parameters (regularizer)}}$$

$\rightarrow$ *Maximising the posterior probability* of the model is equivalent to *minimising costs*.

Probabilities vs. Cost Functions – Example

- Probabilistic formulation:

maximise $\longrightarrow$ $$P(H \mid D) \sim \prod_{data\ d} e^{-(y_d - f(x_d, w))^2} \cdot e^{-w^2}$$

Gaussian prob. of data deviating from model

Small parameters *w* have larger prior probability

- Formulation as cost function:

$$-\ln(P(H \mid D)) \sim \sum_{data\ d} (y_d - f(x_d, w))^2 + w^2$$

minimise

Square error

Penalty on large *w*

imposes a model bias

Theory of Learning from Data

- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

Using Data

- Use this data to find the best parameters w for each model k

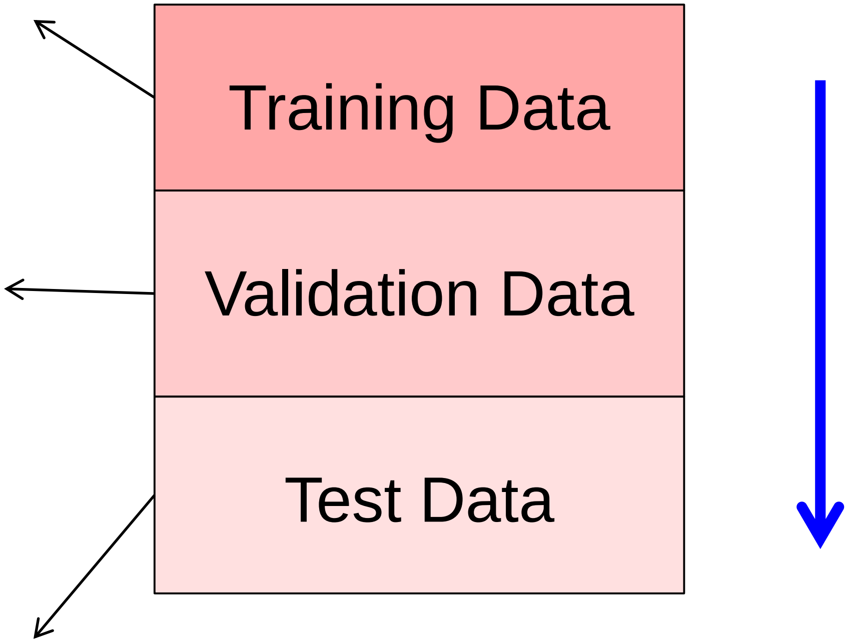
  $$f_k(x, w)$$

- Use this data to calculate an estimate of score $S_k(w)$ for each $f_k(x, w)$ and select

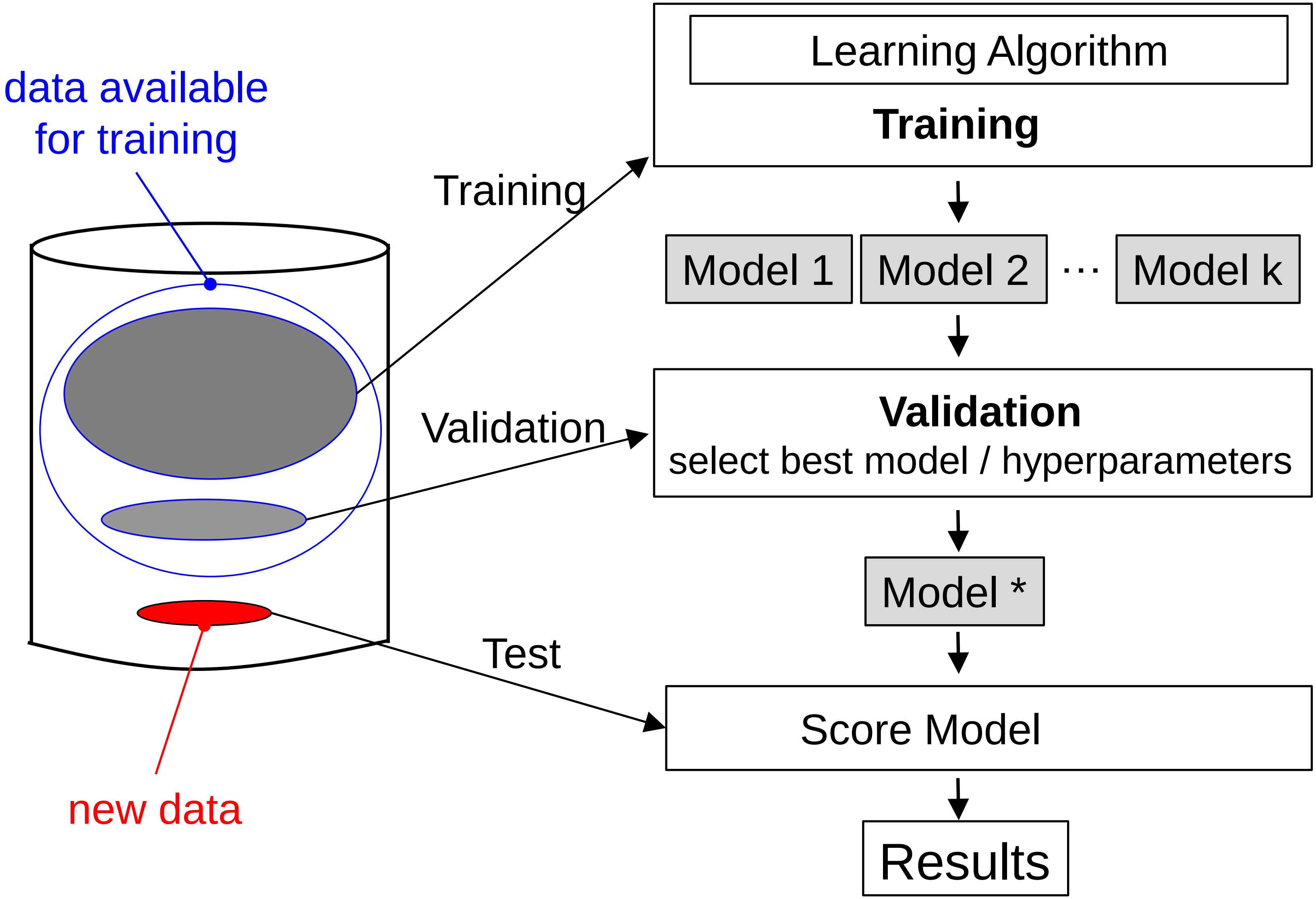
  $$k^* = \mathrm{argmin}_k \, S_k(w)$$

  → *find best hyperparameters*

- Use this data to calculate an unbiased estimate of $S_{k^*}(w)$ for the selected model



The Data Mining Process



Theory of Learning from Data

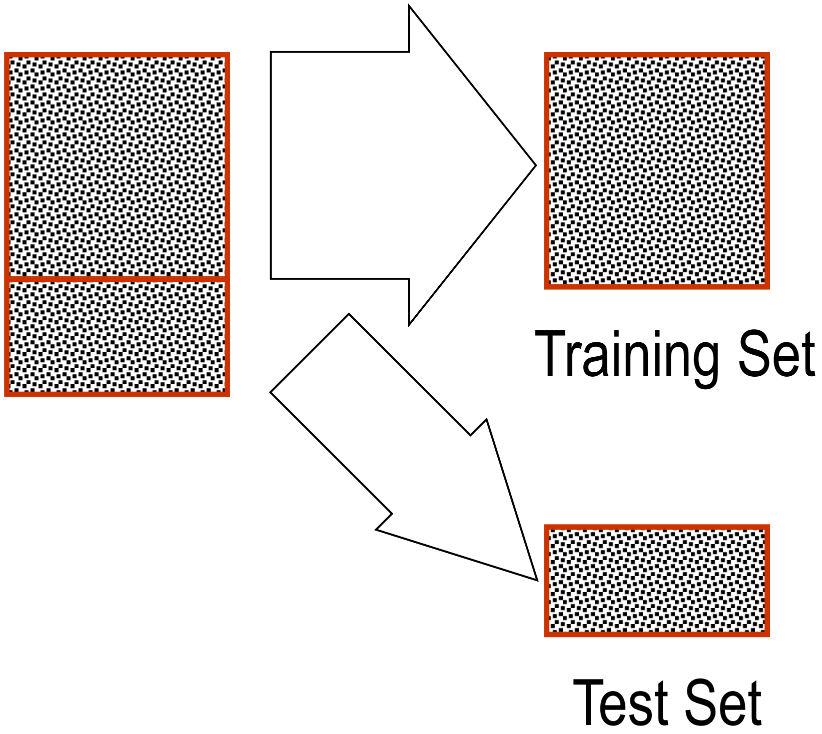
- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

Making Most of the Data

- ***Resubstitution*** method:
  - training data = testing data; naïve strategy, optimistically biased; not for small *n*.
- ***Bootstrap*** method:
  - resample randomly with replacement to generate data sets of same size but different proportion of samples for training and testing.
- ***Holdout*** method:
  - *x*% of data for training , (1-*x*)% for testing.
- ***Rotation*** method (***k*-fold cross validation**):
  - total of *k* data segments, *k*-1 for training, one for testing; repeat *k* times.
- ***Leave-one-out*** method:
  - *n*-1 training samples, one testing sample; repeat *n* times.

Hold-out Method

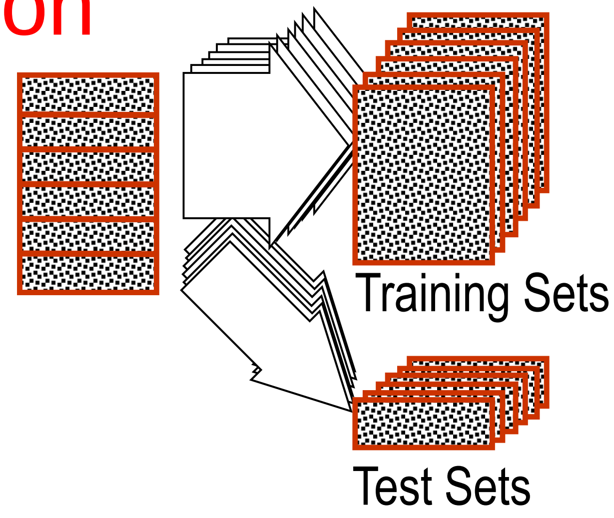
- ***Hold-out set***: Partition data into training and test sets

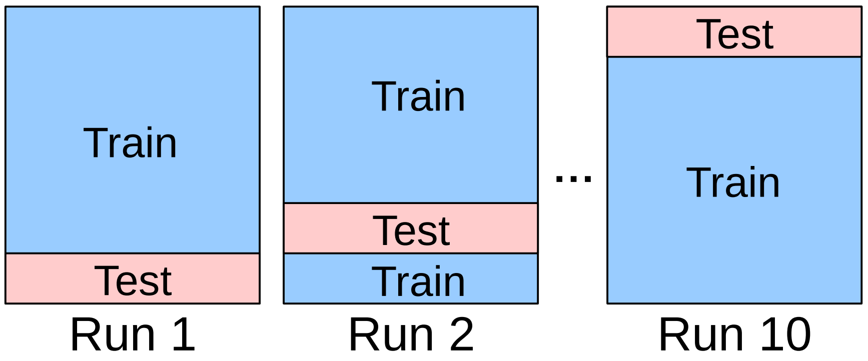


- Data from the test set are "lost" for training
- Different partitioning → different estimates

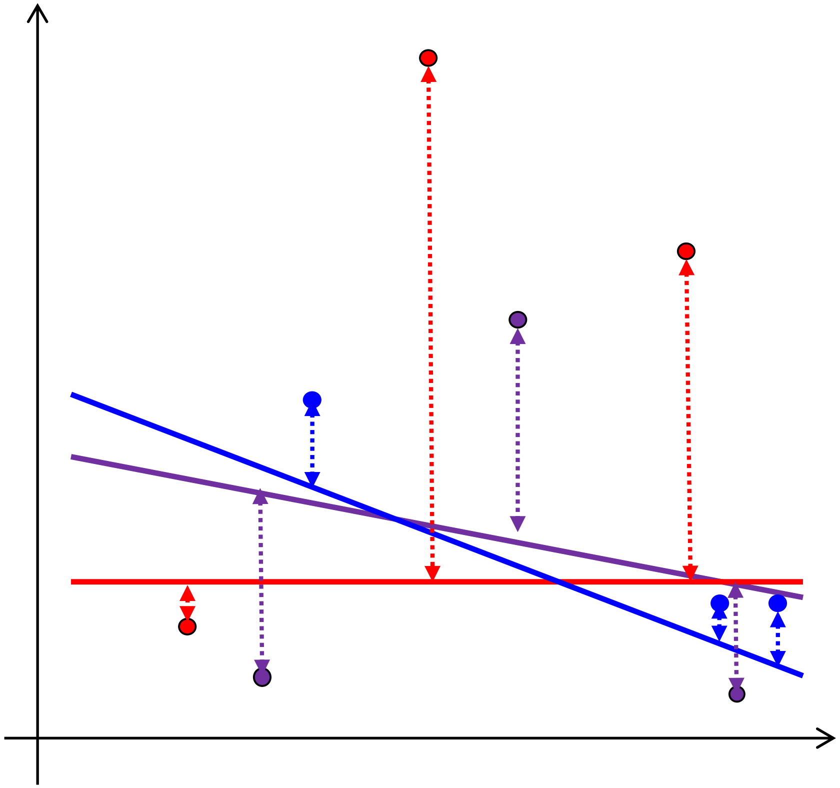
K-fold Cross Validation

- Create K equal partitions
- **Example 1:**
  10-fold cross validation:
  - Use the first 90% of the data set for training and then test on the final 10%
  - Then use the next 10% for testing etc.
- **Example 2:**
  K=n, number of data points
  - “Leave-one-out method”
  - Train n-times with n-1 data points





K-fold Cross Validation (for Regression)



Linear Regression:
$MSE_{3FOLD} = 2.05$

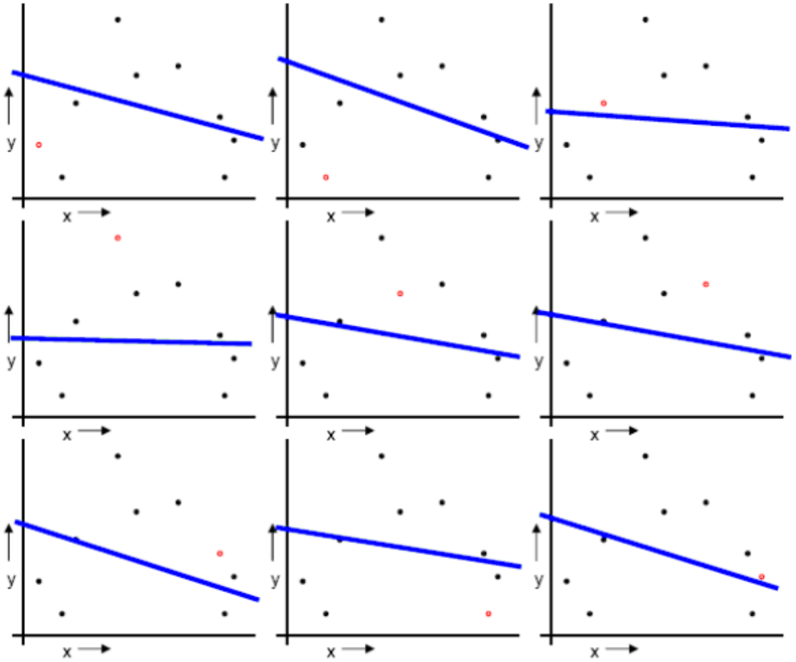
Randomly break the dataset into k partitions
(here: k=3 – red, blue, purple)

- For red=test: Train on the points **not** in the red partition. Find the test-sum of errors on the red points.
- For blue=test: Train on the points **not** in the blue partition. Find the test-sum of errors on the blue points.
- For purple=test: Train on the points **not** in the purple partition. Find the test-sum of errors on the purple points.

Then report the mean square error (MSE).

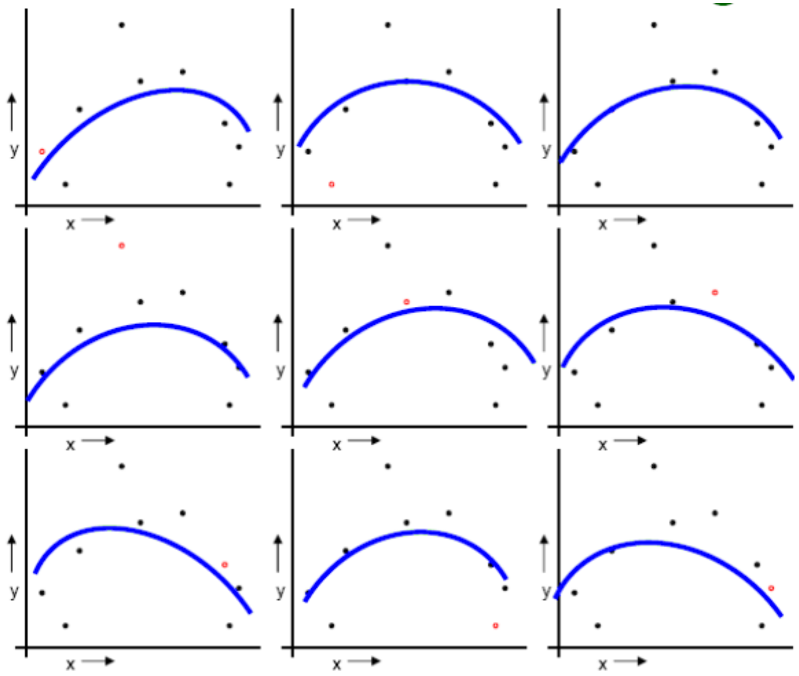
Examples: Leave One Out Cross Validation

Linear regression (2 parameters)



$\mathrm{MSE}_{\mathrm{LOOCV}} = \mathbf{2.12}$

Quadratic regression (3 parameters)



$\mathrm{MSE}_{\mathrm{LOOCV}} = \mathbf{0.962}$

→ quadratic model is better: better hyperparameters

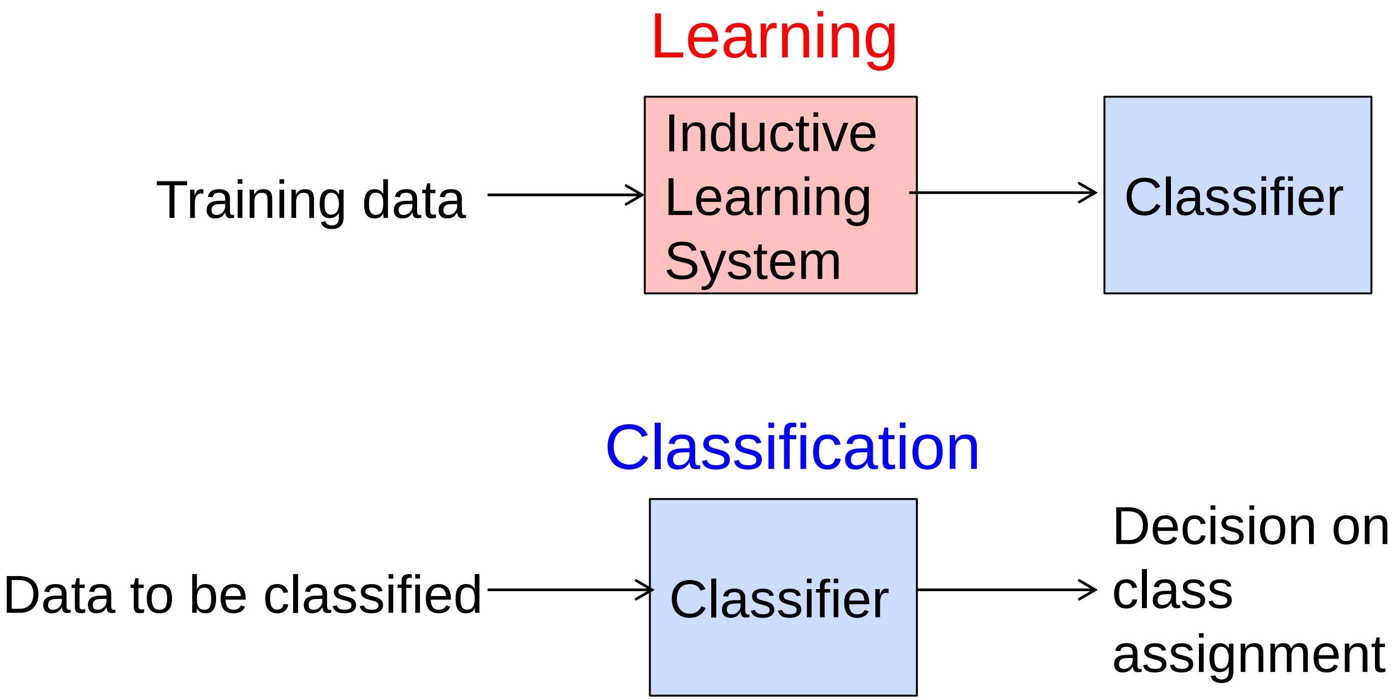
MSE typical for regression.
Which measure for classification?

Theory of Learning from Data

- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

Evaluation of Classification Systems (1)

- Task: Determine which of a fixed set of classes an example belongs to.
- Input: Training set of examples annotated with class values.
- Output: Induced hypothesis (model/concept description/classifier).



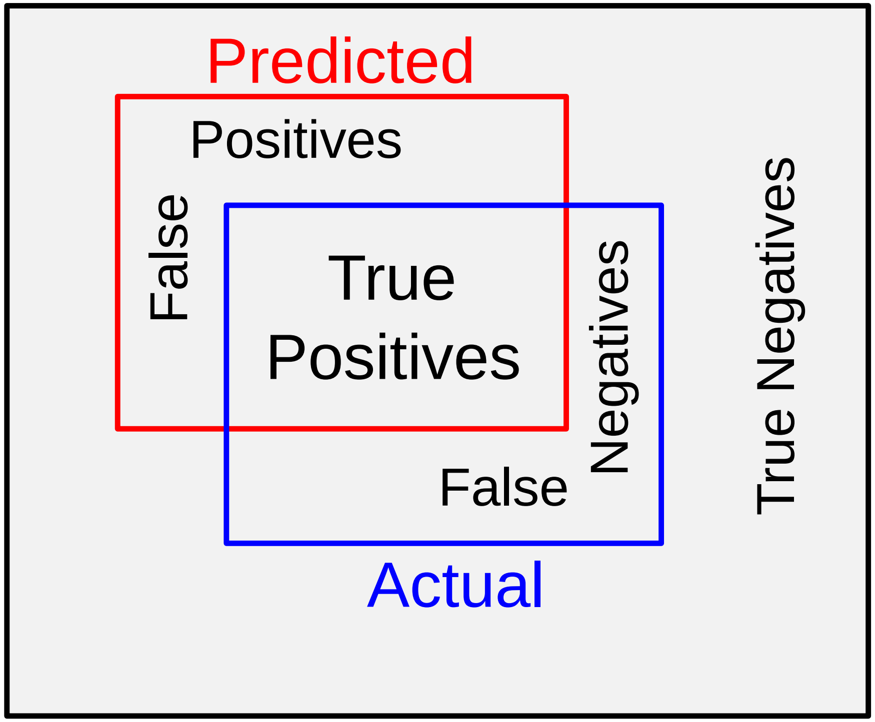
Evaluation of Classification Systems (2)

Evaluation criteria:

- ***Accuracy*** of the classification
- ***Interpretability***
  - E.g. size of a decision tree; insight gained by the user
- ***Efficiency***
  - … of model construction
  - … of model application
- ***Scalability***
  - … for large datasets
- ***Robustness***
  - w.r.t. noise and unknown attribute values

Evaluation of Classification Systems (3)

- Training set: examples with class values for learning.
- Test set: examples with class values for evaluating.
- ***Evaluation***: Model hypotheses are used to classify the test data; results are compared to known classes.



- ***Accuracy***: percentage of examples in the test set that is classified correctly.
- Binary classification: “positive” or “negative”

Theory of Learning from Data

- Model Learning
  - Statistical Learning Theory (VC Dimension, ERM, SRM)
  - Cost Function and its Bayesian View
- Training, Validation & Test Data
  - Cross Validation
- Evaluation of Classification models
  - Confusion Matrix

Classifier Evaluation Metrics: Accuracy & Error Rate

- ***Confusion Matrix***:

Predicted class\Actual class	C_1	$\neg C_1$
C_1	**True Positives (TP)**	**False Positives (FP)**
$\neg C_1$	**False Negatives (FN)**	**True Negatives (TN)**

- ***Classifier Accuracy*, or recognition rate**: percentage of test set tuples that are correctly classified,

$$accuracy\ \ A = \frac{TP + TN}{TP + TN + FP + FN}$$

- ***Error rate***: *1 – accuracy*, or

$$error\ rate = \frac{FP + FN}{TP + TN + FP + FN}$$

Classifier Evaluation Metrics

- ***Sensitivity/Recall***: True Positive recognition rate

=1 if *all data* classified as positive

$$R = \frac{TP}{TP + FN} \qquad (TP + FN = \text{actual positives})$$

- ***Specificity***: True Negative recognition rate

$$SP = \frac{TN}{TN + FP} \qquad (TN + FP = \text{actual negatives})$$

- ***Precision***: exactness – what % of tuples that the classifier labelled as positive are actually positive?

$$P = \frac{TP}{TP + FP}$$

=1 if *just one* data point safely classified as positive

  - Perfect score is 1.0

- Opposing goals when maximising precision & recall

Classifier Evaluation Metrics: *F* Measure

- ***F* measure** (***F_1*** or ***F*-score**): harmonic mean of precision and recall

$$F = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Confusion Matrix / Metrics – Summary

Actual / **Predicted**	Class 1	Class 2
Class 1	True Positive	False Positive
Class 2	False Negative	True Negative

Evaluation metrics:

Accuracy	A = (TP+TN)/(TP+FP+FN+TN)
TP rate, Sensitivity, **Recall**	R = TP/(TP+FN)
FP rate	FPr = FP/(FP+TN) = 1- TN rate
TN rate, Specificity	SP = TN/(FP+TN) = 1 - FPr
Precision	P = TP/(TP+FP)
F-score	F = 2 P ·R / (P+R)

Classifier Evaluation Metrics: Example Confusion Matrix

Actual class\ Predicted class	buy_computer = yes	buy_computer = no	Total	Recognition (%)
buy_computer = yes	**6954**	**412**	7366	99.34 *sensitivity*
buy_computer = no	**46**	**2588**	2634	86.27 *specificity*
Total	7000	3000	10000	95.42 *accuracy*

- Given m classes, an entry, $\boldsymbol{CM_{i,j}}$ in a ***confusion matrix*** indicates # of tuples in class ***i*** that were labeled by the classifier as class ***j***.
- Extra rows/columns may provide totals or recognition rate per class.

Confusion Matrix for Three Classes

True Class				
Classification Model	0	1	2	Total
0	28	**1**	**4**	33
1	**2**	28	**2**	32
2	**0**	**1**	24	25
Total	30	30	30	90

$$\textbf{\textit{Error}} = \frac{\boldsymbol{Sum\ of\ non\ diagonal}}{\boldsymbol{Total}} = 10 / 90 = 0.11 \ (11\%)$$

Accuracy = 1 – Error = 1 - 0.11 = 0.89 (89%)

Accuracy Unsuitable for Skewed Distributions

- Typical *Class Imbalance Problem*: majority in negative class

P\A	C1	C2
C1	0	0
C2	10	90

Accuracy	90/100
Recall (sensitivity)	0/10
Precision	0/0
F-Score	0/0

P\A	C1	C2
C1	3	10
C2	7	80

Accuracy	83/100
Recall	3/10
Precision	3/13 =0.23
F-Score	6/23 =0.261

P\A	C1	C2
C1	8	42
C2	2	48

Accuracy	56/100
Recall	8/10
Precision	8/50 =0.16
F-Score	4/15 =0.267

Cost Matrix

	ACTUAL CLASS		
PREDICED CLASS	*c(i | j)*	Class=Yes	Class=No
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

$c(i \mid j) = c_{ij}$ – Cost of misclassifying class *j* example as class *i*

Total cost function: $$C = \sum_i \sum_j c_{ij} \cdot e_{ij}$$

Computing Cost of Classification

Cost Matrix	Actual Class		
Predicted Class	C(i|j)	+	-
	+	-1	1
	-	100	0

Model M_1	Actual Class		
Predicted Class		+	-
	+	150	60
	-	40	250

Model M_2	Actual Class		
Predicted Class		+	-
	+	180	160
	-	10	150

Accuracy = 80%
Cost = 3910

Accuracy = 66%
Cost = <u>980</u>

Summary

- Statistical learning theory provides a theoretical foundation why a more powerful model isn't always better
  - Parallels between probabilistic (Bayes) and cost function formulations
- Validation and test data may come costly
  - cross validation makes the most of available data
- Confusion matrix and various evaluation metrics
  - accuracy, precision, recall, F-score