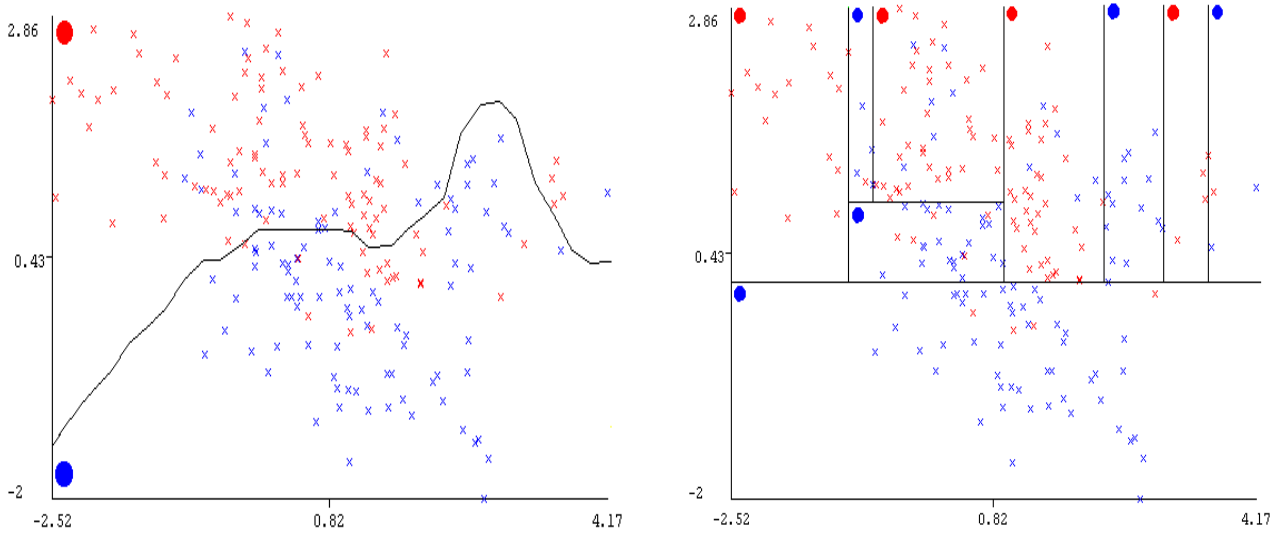
# Evaluation Basics




**Univ.-Lektor Dr.techn. Alexander K. Seewald**
Österreichisches Forschungsinstitut
für Artificial Intelligence


alex@seewald.at / alex.seewald.at

# Overview

**Revisiting State-of-the-art Learning Systems**

- Linear Methods
- Non-Linear Methods from Statistics
- Non-Linear Methods from ML
- Simple & Fast Methods

**Real-Life Example: US Postal Office ZIP Codes**

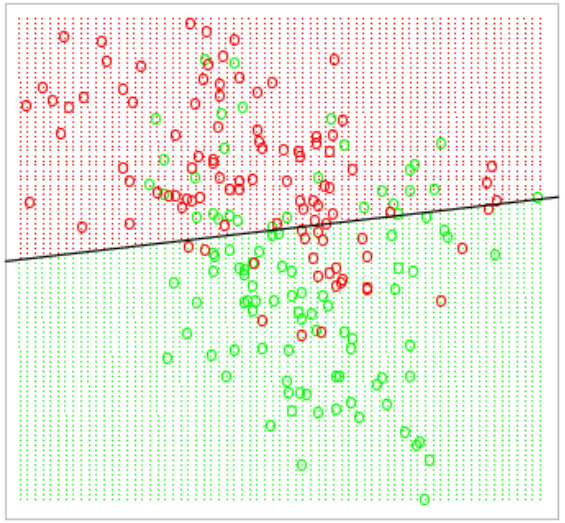
Apply all our learning systems and investigate....

- How to Assess Model Performance? Accuracy and Error
- Approaches to Error Estimation
- Efficient use of sample data: Cross-validation
- Determine significant differences between learners: Fallacies and Pitfalls


alex@seewald.at / alex.seewald.at

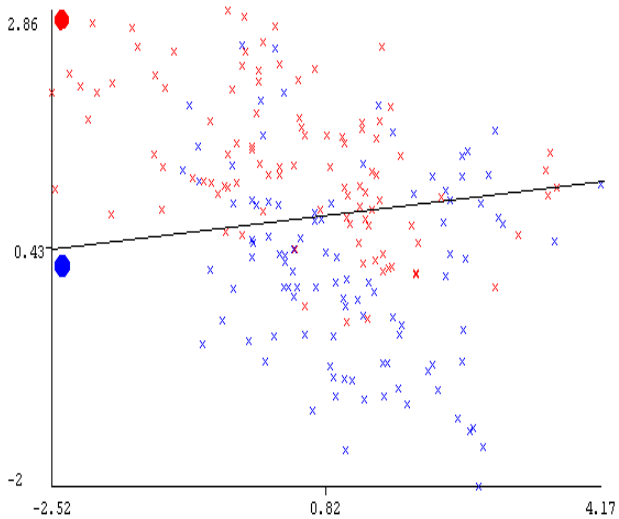
# Linear Methods

All linear methods have high bias and low variance. The decision surface which splits the data into positive and negative example is always a hyperplane (in 2D: a line)



**Linear Regression**

- Minimizes mean squared error
- Very fast, but susceptible to outliers




**Logistic Regression**

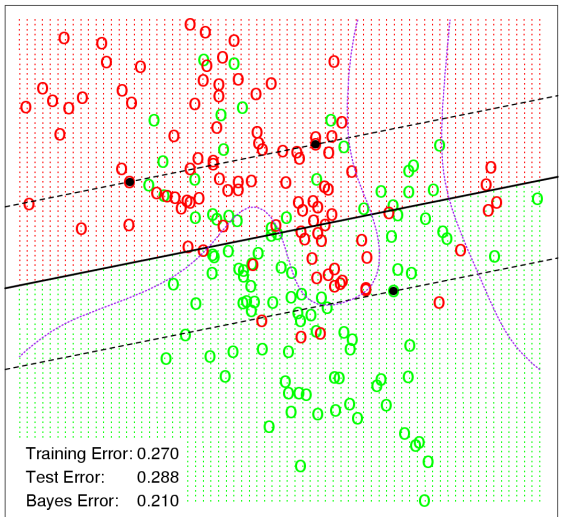
- Regularization: Estimate class probabilities via logistic function, and ensure they sum to 1. Maximizes log-likelihood of model given training data, i.e. *P(f|TD)*
- Quite fast; less susceptible to outliers


alex@seewald.at / alex.seewald.at

# Linear Methods (2)

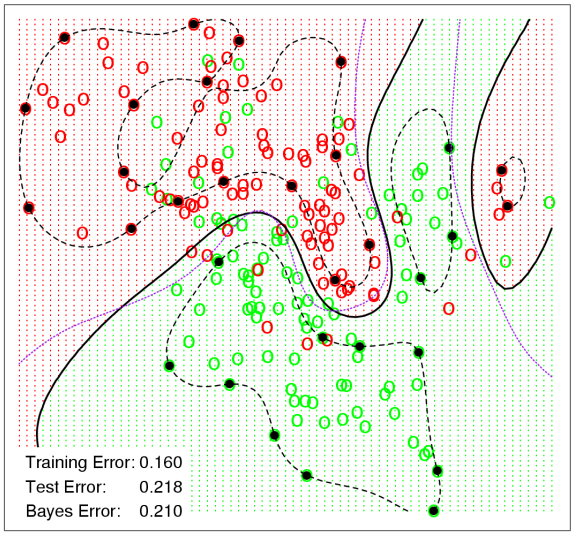
**Support Vector Machine with linear kernel**

- Regularization by defining a well-posed optimization problem: finding the *maximum margin hyperplane* (i.e. maximizing the margin under constraints on misclassifications)
- Fast; least susceptible to outliers
- Similar to logistic regression for two-class tasks.




**Support Vector Machine with nonlinear kernel**

- A linear model in high-dimensional (feature) space defined by a nonlinear kernel. Decision boundary will usually be non-linear in the original feature space.
- Quite fast for polynomial and RBF kernel; slow for complex kernels (e.g. String/Graph kernel)

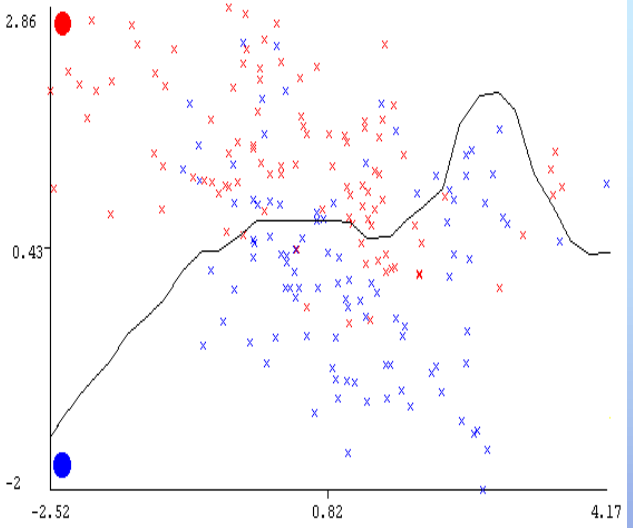




alex@seewald.at / alex.seewald.at

# Non-linear Methods from Statistics

The following non-linear models are low bias and high variance. The decision boundary looks differently in each case and is usually non-linear.

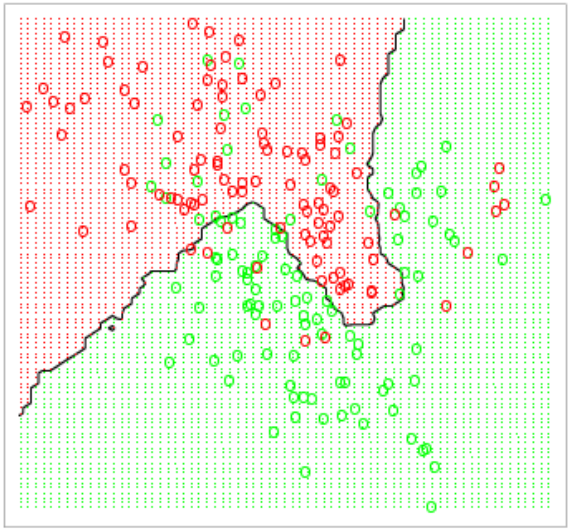



**NaïveBayes**

- Estimates class probabilities directly from TD. Assumes each attribute contributes independently to the final class probablities.
- Very fast. Less suitable for quantitative attributes

**Instance-based learning** (nearest neighbor)

- Classify by similarity with training examples.
- *Universal approximator*: Can learn any concept to abitrary precision given sufficient data
- **But** sufficient data size grows exponentially with the number of input dimensions (*curse-of-dim.*)
- Fast training, slow testing (~$O(N^2)$)

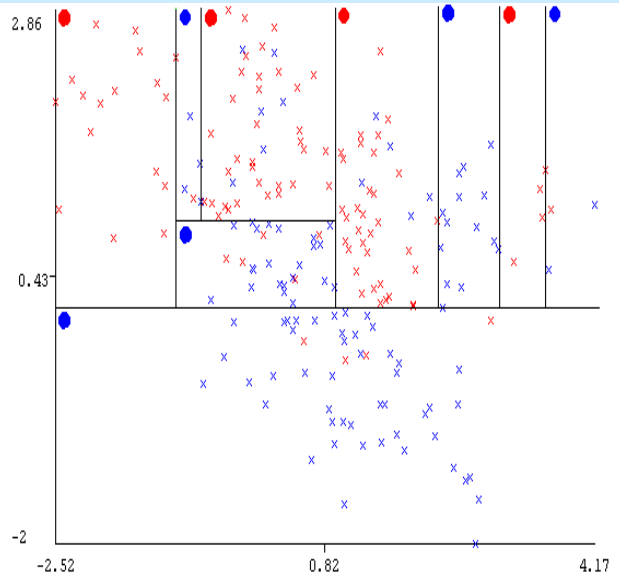



alex@seewald.at / alex.seewald.at

# Non-linear Methods from ML

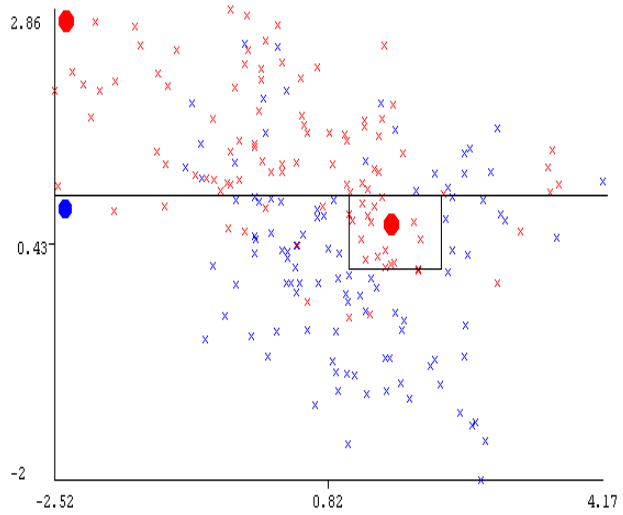
**Decision Trees (C4.5)**

- *Divide-And-Conquer*: Recursive partitioning of training data by attribute values. Creates decision tree with class values at the leaves.
- Only allows axis-parallel splits
- Fast & easy to understand (if tree is small)




**Rule Learning (RIPPER)**

- *Separate-And-Conquer*: Successively partition training data by rules = sets of conditions over attribute values.
- Yields compact and modular descriptions = rule sets. Decision boundaries for each rule are still axis-parallel.
- Slower, but even easier to understand since rules can be analyzed separately.





alex@seewald.at / alex.seewald.at

# Simple & Fast Methods

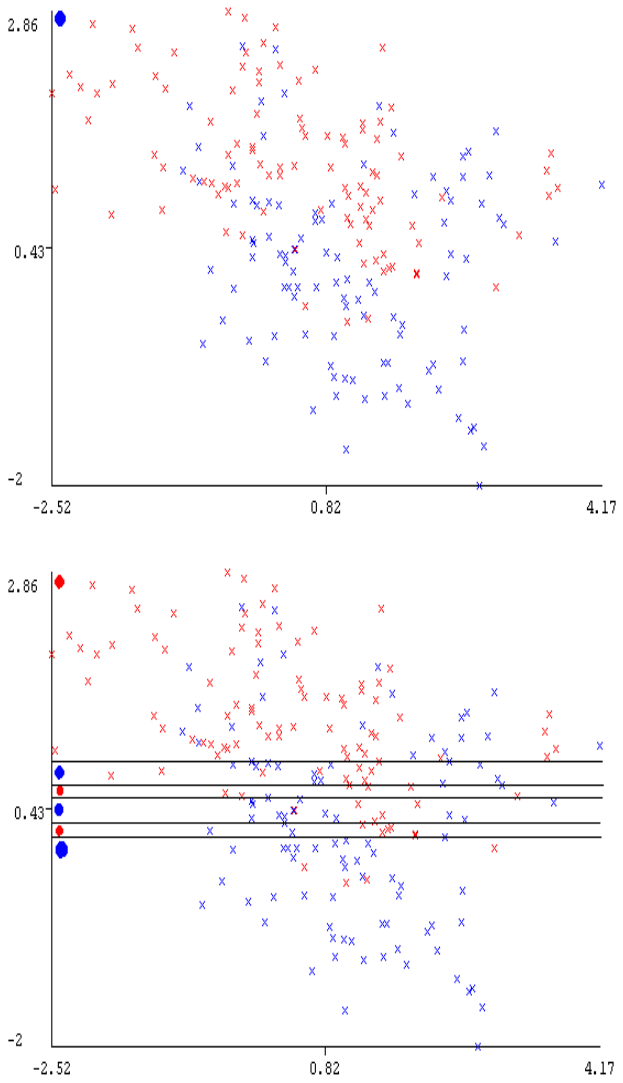
These methods are very fast even for large datasets, and have very high bias & very low variance.

**ZeroR** (gives Baseline Error/Accuracy)

- Predicts most common class from TD, or arithmetic mean for regression tasks
- Measures complexity of learning problem based on class distributions.

**OneR**

- Outputs the best rule based on values of a single attribute.
- Measures complexity of learning problem based on class distributions and the distributions of values from the most predictive attribute.
- Less useful for quantitative attributes.





alex@seewald.at / alex.seewald.at

# An Overview of Learning Systems

| Characteristic | Lin. & Log.R | SVM | Naïve Bayes | Inst. Based | Dec. Trees | Rule Learn. |
|---|---|---|---|---|---|---|
| Natural handling of "mixed" data | – | – | o | – | + | + |
| Handling of MVs | – | – | + | + | + | + |
| Robustness to outliers | –/o | o | o | + | + | + |
| Insensitive to monotone transform. | – | – | – | – | + | + |
| Scalability | + | o/+ | o/+ | – | + | o |
| Robstness concerning irrelevant inputs | – | – | o | – | + | + |
| Interpretability | + | – | o | – | o | + |
| Predictive Power | o | + | – | + | o | o |


alex@seewald.at / alex.seewald.at

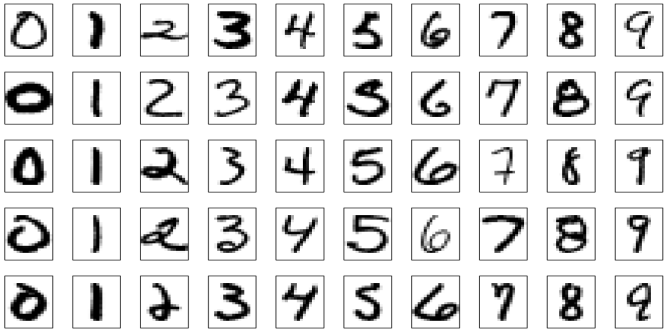
# A Real-Life Example

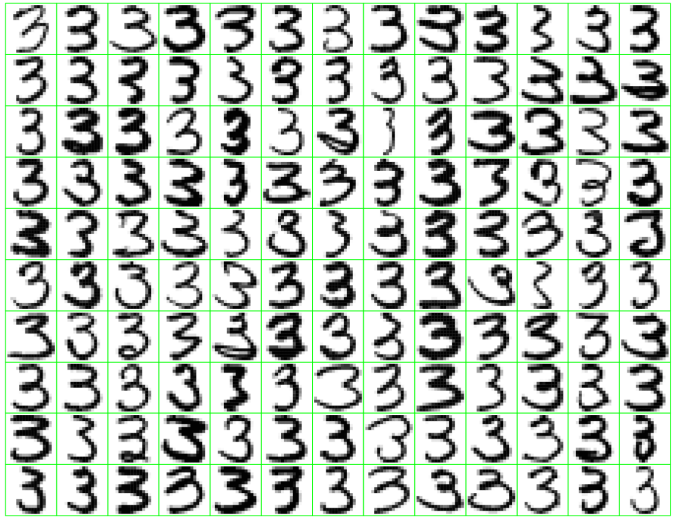
**US Postal Office Digit Dataset**

- Optical Character Recognition for ZIP Codes in the 90ies as a learning task
- Scanned >10,000 digits from more than 500 different people.
- Digits were segmented, resized and resampled to 16x16 pixels with numeric gray values. Each pixel is represented by its own attribute.
- Ten classes: {0,1,2..,9}

⇒ A real-life complex learning task with $p$=256 quantitative attributes, $N$=7291 training example, |C|=10 classes.

**We will apply all our learning methods to this task and see how well they perform. But first...**






alex@seewald.at / alex.seewald.at

# How to Assess Model Performance?

**Contingency table / confusion matrix**

Describes performance of learning system on a dataset with C classes.

CxC Matrix $\mathbf{E}=\{e_{ij}\}$. Entry $e_{ij}$ = number of examples of class *i* for which the learning system predicts class *j*. Obviously, $\Sigma\Sigma\, e_{ij} = |TD| = N$.

**Most widespread measure for Model Assessment:**

*Accuracy* = $\Sigma$diag(**E**)/N (% of correct predictions)

*Error* = 1-*Accuracy* (% of incorrect predictions)

Predicted class

| *0* | *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | |
|---|---|---|---|---|---|---|---|---|---|---|
| **355** | 0 | 2 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | *0* |
| 0 | **255** | 0 | 0 | 6 | 0 | 2 | 1 | 0 | 0 | *1* |
| 6 | 1 | **183** | 2 | 1 | 0 | 0 | 2 | 3 | 0 | *2* |
| 3 | 0 | 2 | **154** | 0 | 5 | 0 | 0 | 0 | 2 | *3* |
| 0 | 3 | 1 | 0 | **182** | 1 | 2 | 2 | 1 | 8 | *4* |
| 2 | 1 | 2 | 4 | 0 | **145** | 2 | 0 | 3 | 1 | *5* |
| 0 | 0 | 1 | 0 | 2 | 3 | **164** | 0 | 0 | 0 | *6* |
| 0 | 1 | 1 | 1 | 4 | 0 | 0 | **139** | 0 | 1 | *7* |
| 5 | 0 | 1 | 6 | 1 | 1 | 0 | 1 | **148** | 3 | *8* |
| 0 | 0 | 1 | 0 | 2 | 0 | 0 | 4 | 1 | **169** | *9* |

True class

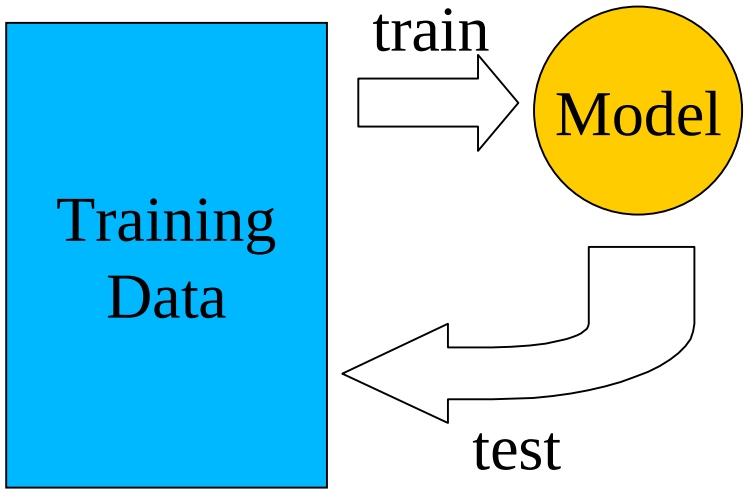
Example: N=2007,
C=10 classes (0-9)
$\Sigma$diag(**E**) = **1894**
$\Rightarrow$ *Accuracy* 94.4%
$\Rightarrow$ *Error* 5.6%


alex@seewald.at / alex.seewald.at

# Resubstitution Estimate

**Training Set Error:** Error of model on training data.

| | | |
|---|---|---|
| ZeroR (Baseline) | 83.62% | |
| OneR | 64.33% | |
| Naïve Bayes | 23.03% | |
| Linear Regression | 7.60% | |
| RIPPER (Rule Learning) | 4.66% | |
| C4.5 (Decision Tree L.) | 1.98% | |
| SVM w/ linear kernel | 0.15% | ($d$=1, $c$=0, $\lambda$=1) |
| SVM w/ poly. kernel | 0.01% | ($d$=5, $c$=0, $\lambda$=10) |
| Logistic Regression | 0.00% | |
| IB1 (Instance-Based L.) | 0.00% | |





alex@seewald.at / alex.seewald.at

# Analytical Error Estimation

***Training Set Error*** **is usually too optimistic, and can be misleading for some learning systems** (e.g. IB1: always 100%). **It estimates how well the data can be approximated by a given model, but does not yield a good estimate of true error = error on previously unseen data.**

For some learning methods a useful error estimate can be derived analytically just from the training set:

Linear Regression : $$Error = Err_{resubst.} + 2\frac{p+1}{N}\sigma_\varepsilon^2$$

where $\sigma_\varepsilon^2 = \sum_{i=1}^{N} \frac{(y_i - f(\mathbf{x_i}))^2}{N-1}$ is the sample variance of the residual squared error

Logistic Regression (2 classes) : $$AIC = -2\ell(\beta) + 2\frac{p+1}{N}$$

However, for most learning methods this does not work well.

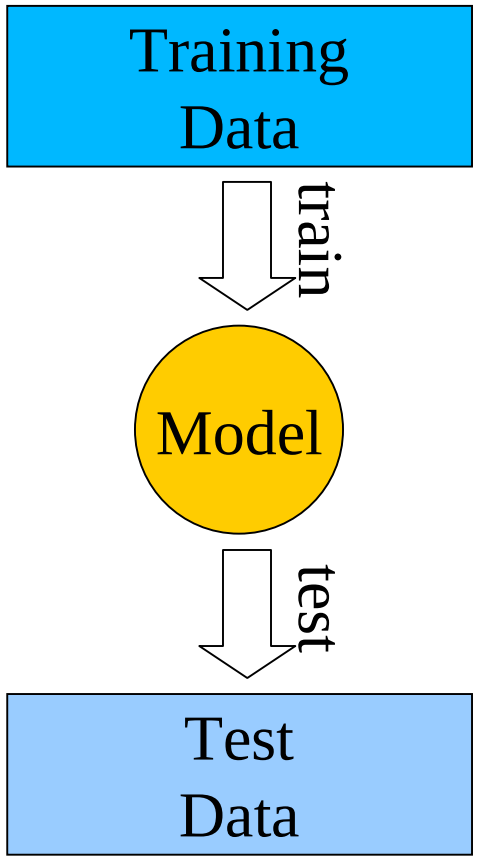

alex@seewald.at / alex.seewald.at

# Hold-out Set Estimate

**Test Set Error:** Error of model on *independent* test data

*(i.e. not used for training)*

| | |
|---|---|
| ZeroR (Baseline) | 82.11% |
| OneR | 68.56% |
| Naïve Bayes | 28.70% |
| RIPPER (Rule Learning) | 16.64% |
| C4.5 (Decision Tree L.) | 15.00% |
| Linear Regression | 13.05% |
| Logistic Regression | 10.91% |
| SVM w/ linear kernel | 7.08% |
| IB1 (Instance-Based L.) | 5.63% |
| SVM w/ poly. kernel | 4.29% |





alex@seewald.at / alex.seewald.at

# Hold-out Set Estimate (2)

**Repeated hold-out testing**

- Compute *Hold-out Set Estimate* several times with differently shuffled training data which is randomly split into new training and test sets. Determine average and standard deviation of obtained errors/accuracies to estimate expected performance and its variance.

**Variant: 0.632 Bootstrap**

- Sample from training data *with replacement* to get training set of size N. Use remaining data for testing.
- $Error = 0.632 Err_{test} + 0.368 Err_{resubst.}$ Estimate does not work well for models that have overfitted the training data. ($Err_{resubst.} << Err_{test}$)


alex@seewald.at / alex.seewald.at

# Observations

*Test Set Error* is far more accurate than *Training Set Error*, but needs to *hold out* a significant part of data from the training set (~25-50%) as an independent test set. This is unsatisfactory: the additional data could be used to build a better model.

*Repeated hold-out testing* computes better expected errors, and also estimates the expected error variance. However, all of the test data is still lost for training.

***Crossvalidation* solves this problem and makes it possible to use almost all data for training, while still computing a useful error estimate - at additional computational cost.**

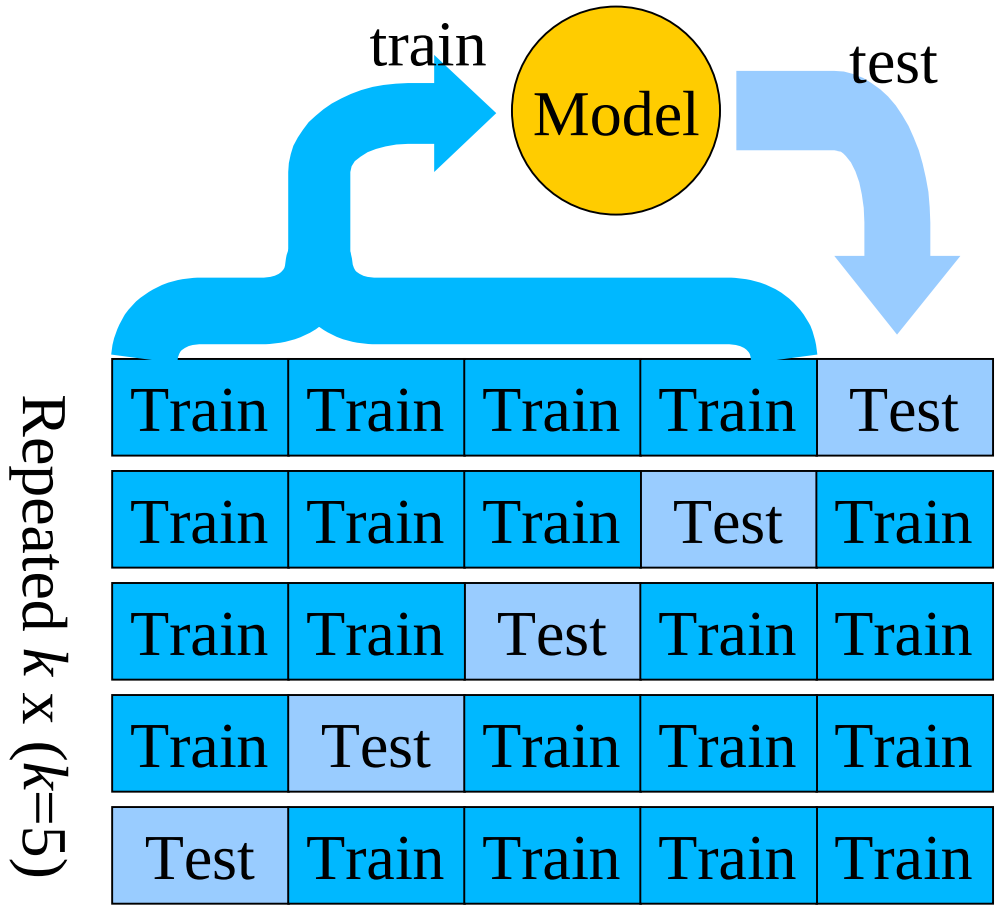

alex@seewald.at / alex.seewald.at

# Crossvalidation Estimate

**Crossvalidation**: Split training data into *k* equal-sized folds; use one fold for testing and all others for training.

(*k*=10 below)

| | |
|---|---|
| ZeroR | 83.62% |
| OneR | 69.41% |
| Naïve Bayes | 25.87% |
| RIPPER | 12.49% |
| C4.5 | 11.69% |
| Linear Reg. | 9.29% |
| Logistic Reg. | 8.50% |
| SVM linear | 4.06% |
| IB1 | 2.96% |
| SVM poly. | 1.73% |





alex@seewald.at / alex.seewald.at

# Crossvalidation Estimate (2)

**Observations**

- As with *Repeated Hold-out testing*, Crossvalidation is computationally costly: The learning system is trained $k$ times with ($k$-1)/k and tested on 1/$k$ of the full data.
- However, less data is lost. E.g. for $k$=10, 90% of data is used for training, and only 10% is needed for testing. Still, each part of the data is used for testing exactly once, while the training sets heavily overlap.
- Most common accuracy/error estimation within ML & DM ($k$=5 or 10)

**Variants of CV**

- *Stratified CV*: Ensure the same class distributions in each fold as in the full training data. Introduces some bias into the sampling, but reduces variance.
- *Leave-one-out CV*: Crossvalidation with $k = N$ = number of examples, so that each fold contains only a single example. This is almost unbiased, but may have high variance. By definition *leave-one-out* cannot be stratified, so there is no easy way to reduce the variance. Computationally *very* costly. For some classifiers, *leave-one-out* can be computed much faster (e.g. SVM, IB & LinR)


alex@seewald.at / alex.seewald.at

# Determining Significant Differences

**To compare two learning systems, a statistical significance test is needed.**

**Each test has the following properties:**

- *Power* (probability to find a significant difference that is really there)
- *Type I (alpha) error* (prob. to find a significant difference when there is none)
- *Type II (beta) error* (1-*Power*; prob. to overlook a real significant difference)

**General testing procedure**

- The null hypothesis is that the two algorithms perform similarly (i.e. no significant difference). Running the experiments, computing the test statistic and determining the *p*-value gives us the probability that the null hypothesis is right – given that the test's assumptions are correct.
- If *p*-value < *significance level* (e.g. 5%), then we reject the null hypothesis and assume that there is a significant difference between the two algorithms.

**To compare many learning systems: Analysis of Variance (ANOVA). Repeated significance tests are best avoided, because of alpha error: significance level of 5% means that to compare our 10 algorithms against each other (45 comparisons) we expect 2-3 spurious significant differences.**


alex@seewald.at / alex.seewald.at

# Significance Tests

**For a single *k* fold CV** (Algorithm A vs. Algorithm B)

- *$X^2$ Test after McNemar:* Compute (pseudo) confusion matrix with the correctness of A's prediction as rows and B's prediction as columns. A+, B+: correct prediction. A-,B-: incorrect prediction. Degrees-of-freedom (df) = 1.

$$X^2 = \frac{(c-b)^2}{c+b}$$

| | B+ | B- |
|---|---|---|
| A+ | a | **b** |
| A- | **c** | d |

- *Paired (Student) t-Test:* Compute differences Diff=$Err_A$-$Err_B$ for each fold separately. The average of the values should be large relative to the standard deviation to reject null hypothesis. Significant values of *t* depend on degrees-of-freedom (df) and chosen significance level.

$$\overline{Diff} = \frac{1}{k}\sum_{i=1}^{k} Diff_i \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{k}\left(Diff_i - \overline{Diff}\right)^2}{k-1}}$$

$$t = \frac{\overline{Diff}}{\sigma}\sqrt{df+1} \qquad df = k-1$$

*df may be overestimated by ~ 50% since training folds are not independent.*


alex@seewald.at / alex.seewald.at

# Significance Tests (2)

**For multiple *k* fold CV**

- *Most common approach:* **10x 10-fold CV** (each learning system gets the same train/test folds) and paired t-Test. I.e. a *Paired t-Test* with $Err_A$ and $Err_B$ computed over all ten folds from each single CV. $k$=10=number of runs, $df$=9
  *df may be overestimated since runs are not independent ⇒ higher alpha error*

- Previously proposed [Dietterich, 1998]: **5x 2-fold CV** as an alternative to 10x 10fold CV. Procedure is same as above, i.e. averaged error over folds, but uses a two-fold CV plus five reptitions. This has low alpha error, but also high beta error, which translates to low power.

- Using all 100 error estimates from 10x 10fold CV. I.e. a *Paired t-Test* with $Err_A$ and $Err_B$ from each fold treated separately as independent estimate. $k$=100, so theroretically this would mean $df$=99, but since there is so much redundancy, $df$=10 has been proposed for binary data based on experiments with synthetic data [Bouckaert, 2003]. Has slightly higher *replicability* (repeating the experiments yields more similar results on average), but estimation of effective *df* for non-binary data is still undergoing investigation.


alex@seewald.at / alex.seewald.at