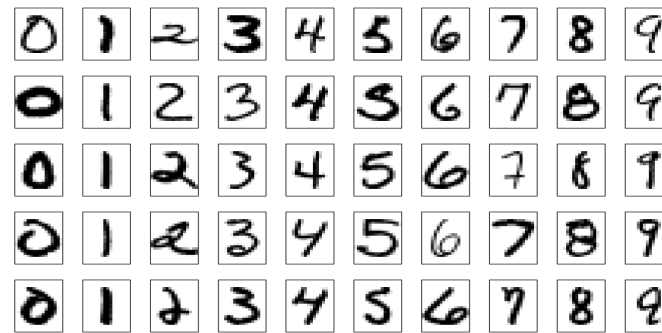
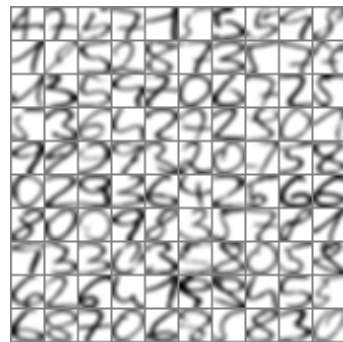


Evaluation Basics



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

Evaluation Basics

**Before interpreting Machine Learning results,
it is essential that the learned models are evaluated.**

- *Evaluation* means assessing model performance, i.e. verifying how good the learned model fits available data.
- Only models which fit the available data sufficiently well can be sensibly interpreted.
- There is always a risk of interpreting specific characteristics (quirks ;-) of the machine learning model rather than characteristics of the underlying data.

Evaluation, if properly done, can minimize this risk.

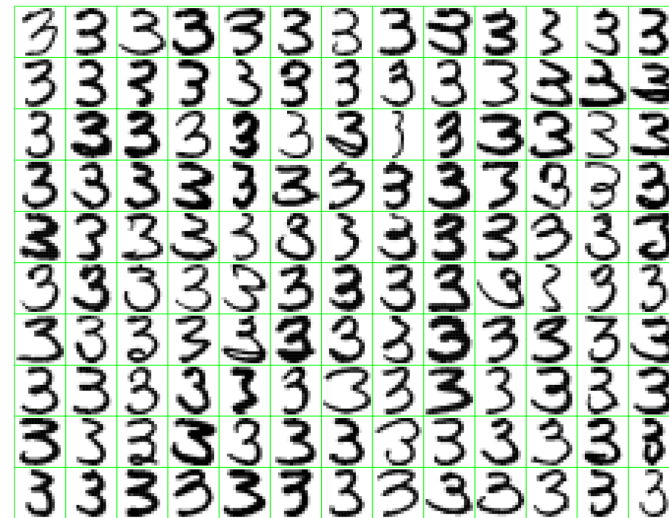
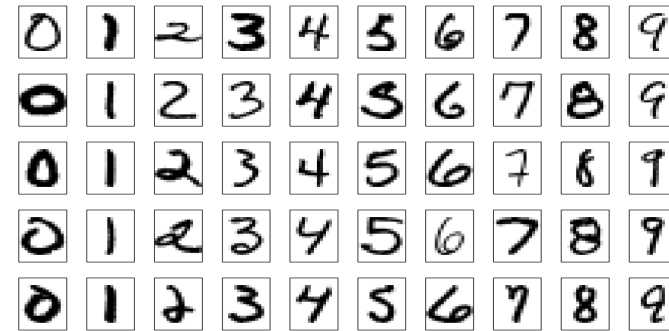
- **Most important principle:** No information must flow from test set into training, even indirectly! Even running too many experiments on the same data can be dangerous.

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...



How to Assess Model Performance?

Contingency table / confusion matrix

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

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

Most widespread measure for Model Assessment:

Accuracy = $\sum \text{diag}(\mathbf{E}) / N$ (% of correct predictions)

Error = $1 - \text{Accuracy}$ (% 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

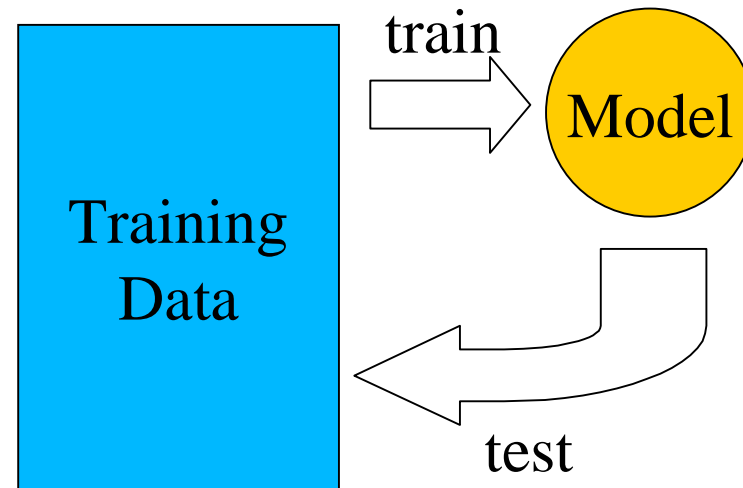
Example: $N=2007$,
 $C=10$ classes (0-9)
 $\sum \text{diag}(\mathbf{E}) = 1894$
 $\Rightarrow \text{Accuracy } 94.4\%$
 $\Rightarrow \text{Error } 5.6\%$

True class

Resubstitution Estimate

Training Set Error: Error of model on training data.

ZeroR (Baseline)	83.62%
<u>OneR</u>	<u>64.33%</u>
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%



Analytical Error Estimation

Training Set Error is usually too optimistic, and can be misleading for some learning systems (e.g. IB1: always 0%). 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):

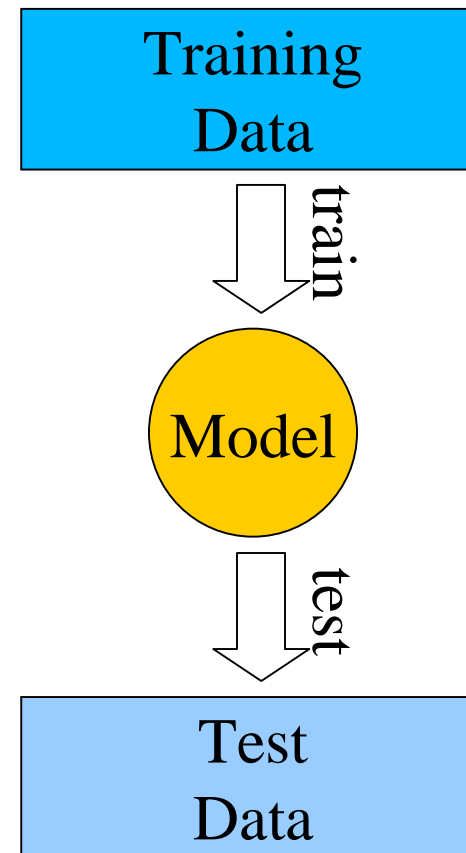
$$AIC = -2\ell(\beta) + 2 \frac{p+1}{N}$$

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

Hold-out Set Estimate

Test Set Error: Error of model on *independent* test data
(*i.e. not used for training*)

ZeroR (Baseline)	82.11%
<u>OneR</u>	<u>68.56%</u>
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%



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.
- $Error = 0.632Err_{test} + 0.368Err_{resubst.}$ Estimate does not work well for models that have overfitted the training data. ($Err_{resubst.} \ll Err_{test}$)

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.

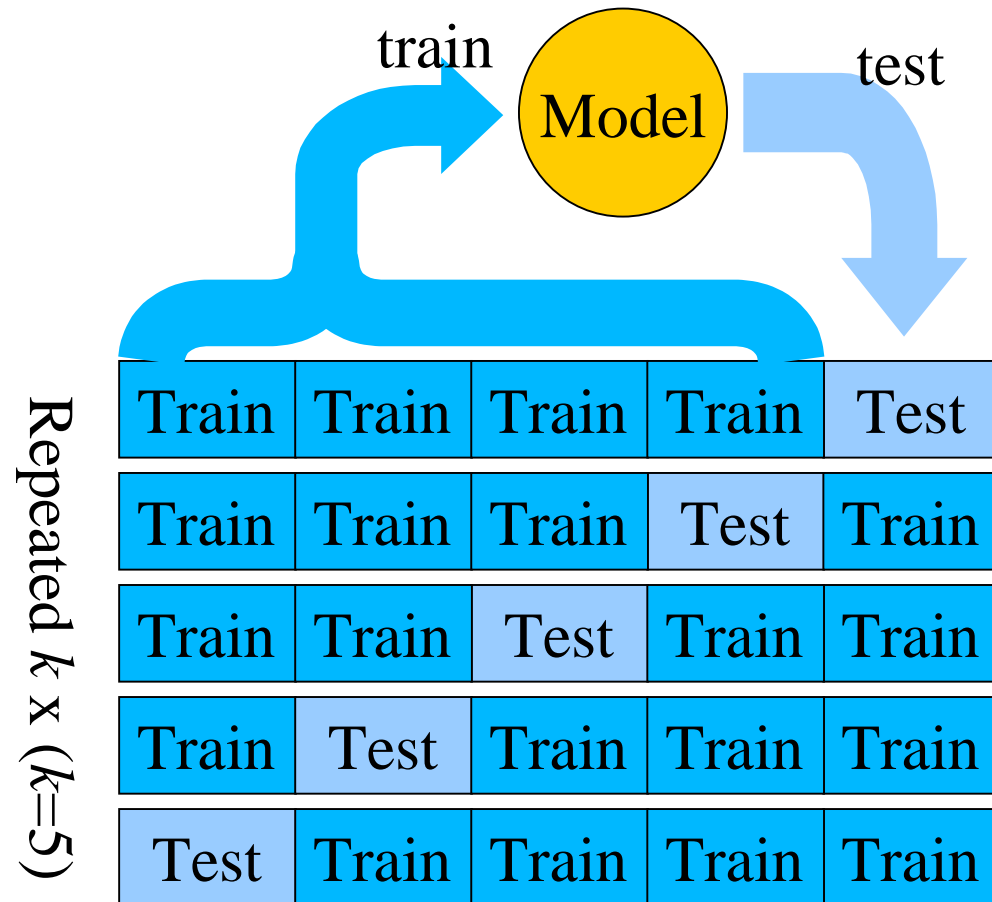
***Crossvalidation* solves this problem and makes it possible to use almost all data for training, while still computing a useful error estimate - at additional computational cost.**

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%
<u>OneR</u>	<u>69.41%</u>
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%



**WEKA reports
this by default**

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. WEKA uses this with $k=10$.

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)

Observations

As with *Repeated Hold-out testing*, Crossvalidation is computationally costly: The learning system is trained k times with $(k-1)/k$ of the full data 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.

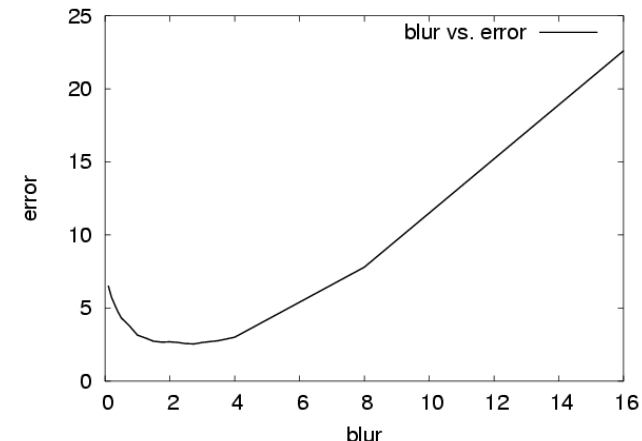
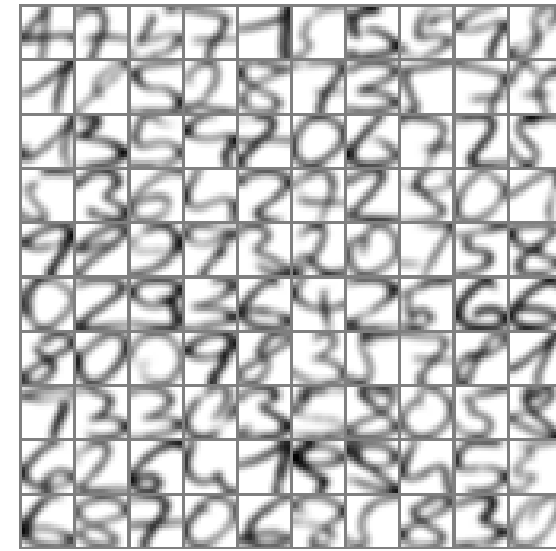
Most common accuracy/error estimation within ML & DM ($k=5$ or 10)

Our Digit Dataset

Our very own Digit Dataset

- 4,989 digits contributed by 50 people.
- Array lines were found via vertical/horizontal histograms and refined via local search and linear regression. Digits were segmented and arbitrarily resized (ignoring aspect ratio) to 16x16 pixels with anti-aliasing. Each pixel is represented by an attribute.
- Classes {0,1,2...,9}, equally represented.
- Blurring the digits improves error rate by reducing differences (see bottom right)

Error rate comparable to USPS, but model is not transferrable either way (i.e. train USPS, test our digits and vice versa both fail with ~ 50% error)



Preprocessing Methods for Digits

Within WEKA

- weka.attributeSelection.PrincipalComponents

PCA can reduce the dimensionality of the dataset. Should be done only on training and used for test set when used for preprocessing. However this is not well supported.

- weka.filters.unsupervised.attribute.AddExpression

Horizontal/vertical histograms can be computed by summing over the sixteen column / row pixels. However this is a quite cumbersome and limited approach.

Fortunately, performance is quite good even without further preprocessing.

Preprocessing Methods for Digits (2)

Outside WEKA

- ImageMagick image library for image preprocessing (C, C++, Perl, Tcl/Tk...) - very useful.
- Many features proposed for digit recognition. Surveys:
Cheng-Lin Liu, [Kazuki Nakashima](#), [Hiroshi Sako](#), [Hiromichi Fujisawa](#): *Handwritten digit recognition: benchmarking of state-of-the-art techniques*. [Pattern Recognition](#) 36(10): 2271-2285 (2003).
Loncaric, S., *A survey of shape analysis techniques*, *Pattern Recognition* 31(8):983-1001 (1998).
- Implementing these yourself is usually essential.

If there is interest, I can give these as alternative exercises

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 $p\text{-value} < \text{significance level}$ (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.

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 $\text{Diff} = \text{Err}_A - \text{Err}_B$ 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.

$$\overline{\text{Diff}} = \frac{1}{k} \sum_{i=1}^k \text{Diff}_i$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^k (\text{Diff}_i - \overline{\text{Diff}})^2}{k - 1}}$$

$$t = \frac{\overline{\text{Diff}}}{\sigma} \sqrt{df + 1}$$

$$df = k - 1$$

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

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. Also the only test available within WEKA Experimenter. $k=10$ =number of runs, $df=9$ *df may be overestimated since runs are not independent \Rightarrow 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 repetitions. This has low alpha error, but also high beta error, which translates to low power.

Finalize Group Constitution

For students who have already found a group, but have not yet sent me a mail and received a group number

- Please come forward and announce the group in person now. You will be assigned a new group number.

For students who have not yet found a group

- I will try to build new groups out of these students, and assign the remaining students to smaller groups arbitrarily.

If you represent a group with less than four people

- Stay, so we can make the assignment of remaining students to groups less arbitrary. ;-)

This should be over in a few minutes, so bear with me.