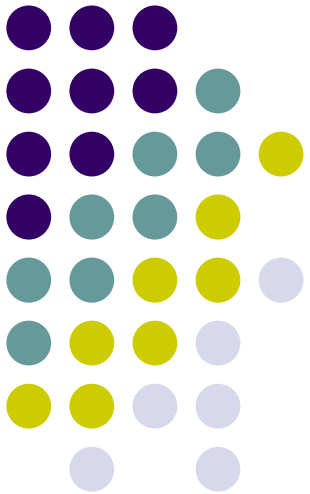
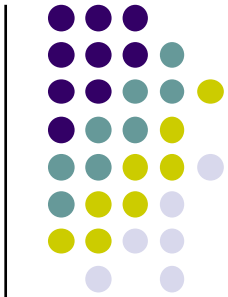
Evaluation Methodology

J. Savoy
Université de Neuchâtel



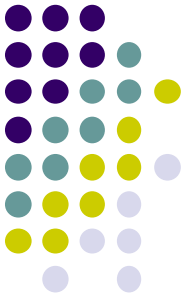
Ian H. Witten, Eibe Frank, M. A: Hall: Data Mining. Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

Overview



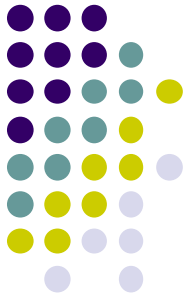
- Issues: training, testing, tuning
- Predicting performance: confidence limits
- Holdout, cross-validation, bootstrap
- Comparing schemes: the *t*-test
- Predicting probabilities: loss functions
- Cost-sensitive measures
- Evaluating numeric prediction
- The Minimum Description Length (MDL) principle

Evaluation: the Key to Success



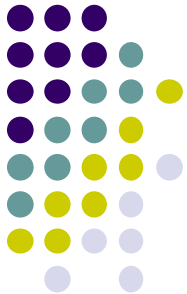
- How predictive is the model we learned?
- Error on the training data is *not* a very good indicator of performance on future data
  - Otherwise 1-NN (or table look-up) would be the optimum classifier!
- Simple evaluation approaches that can be used if a lot of (labeled) data is available (e.g., simulation)
  - Usually split data into training and test sets
- However: (labeled) data is usually limited
  - More sophisticated techniques need to be used
  - What is the cost to obtain more labeled data?

Issues in Evaluation



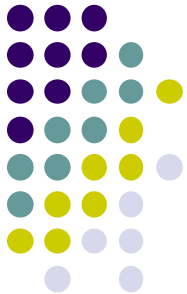
- Statistical reliability of estimated differences in performance (→ significance tests)
  Is a performance difference between 85.8 vs. 85.6 really significant (or due to random factors)?
- Choice of performance measure:
  - Number of correct classifications
  - Accuracy of probability estimates (e.g., naïve Bayes)
  - Error in numeric predictions (e.g., regression)
- Costs assigned to different types of errors
  - Many practical applications involve costs
- The Italian seismologists in 2012 (6 years in jail!)

Training and Testing



- Natural performance measure for classification problems: *error rate*
  - *Success*: instance's class is predicted correctly
  - *Error*: instance's class is predicted incorrectly
  - Error rate: proportion of errors made over the whole set of instances
- *Resubstitution error (apparent error):* error rate obtained from training data
- Resubstitution error is (hopelessly) optimistic!

Training and Testing

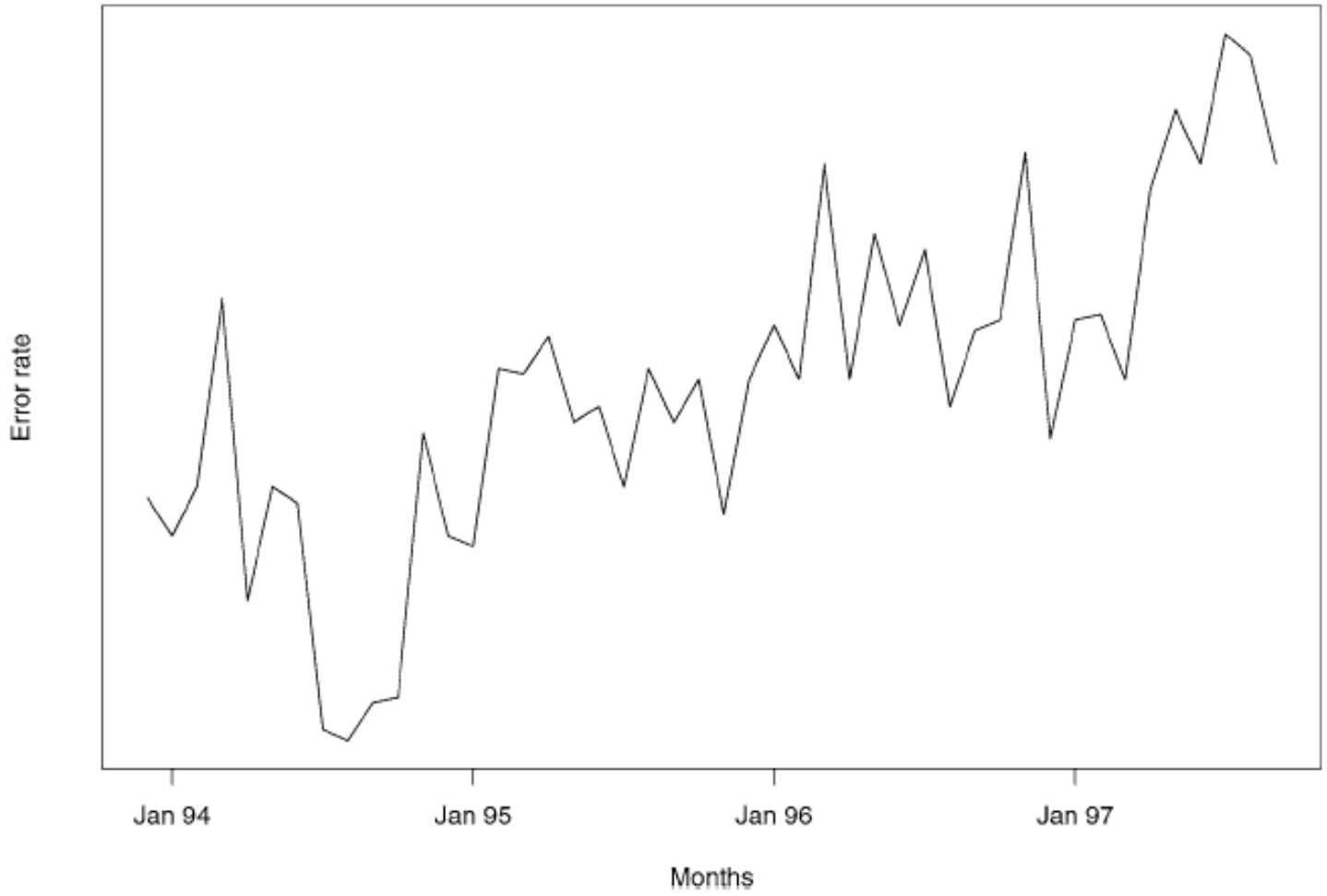


- *Test set*: independent instances that have played no part in formation of classifier
  - Assumption: both training data and test data are representative samples of the underlying problem
- Test and training data may differ in nature
  - Example: classifiers built using customer data from two different towns *A* and *B*

    To estimate performance of classifier from town *A* in completely new town *B*

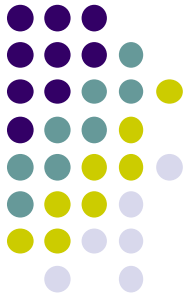
    Thus both samples must *representative*
    But the evolution may change this factor!

The evolution (of the error rate)



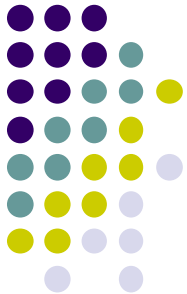
Hand, D.J.. (2006). Classifier Technology and the Illusion of Progress. *Statistical Science, 21(1), 1-14*.

Parameter Tuning



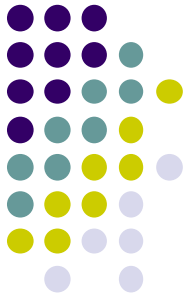
- It is important that the test data is not used *in any way* to create the classifier
- Some learning schemes operate in two stages:
  - Stage 1: build the basic structure (training set)
  - Stage 2: optimize parameter settings / do some selections (validation set)
- The test data can't be used for parameter tuning! Trump's philosophy: build walls
- In this case, proper procedure uses *three* sets: *training data*, *validation data*, and *test data*
- How much in each case?

Making the Most of the Data



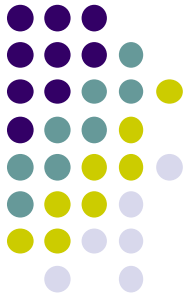
- Once evaluation is complete, *all the data* can be used to build the final classifier
- Generally, the larger the training data the better the classifier (but returns diminish, at some point the gain is minimal)
- The larger the test data the more accurate the error estimate
- *Holdout* procedure: method of splitting original data into training and test set
  - Dilemma: ideally both training set *and* test set should be large!

Predicting Performance



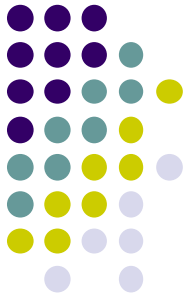
- Assume the estimated error rate is 25%. How close is this to the true error rate?
  - Depends on the amount of test data
- Prediction is just like tossing a (biased!) coin
  - head is a *success*, tail is an *error*
- In statistics, a succession of independent events like this is called a *Bernoulli process*
  - Statistical theory provides us with confidence intervals for the true underlying proportion

Confidence Intervals



- We can say: *p (true proportion)* lies within a certain specified interval with a certain specified confidence
- Example: *S* = 750 successes in *N* = **1,000** trials
  - Estimated success rate: 75%
  - How close is this to true success rate *p*?
  - Answer: with 80% confidence *p* in [73.2, 76.7]
- Another example: *S* = 75 and *N* = **100**
  - Estimated success rate: 75%
  - With 80% confidence *p* in [69.1, 80.1]
- Increasing N reduces the uncertainty

Mean and Variance



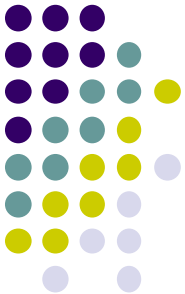
- Mean and variance for a Bernoulli trial: p and $p \cdot (1-p)$
- Expected success rate (estimation) $f = S/N$
- Mean and variance for f : p and $p \cdot (1-p)/N$
- For large enough N, f follows a Normal distribution
- c% confidence interval $[-z \leq X \leq z]$ for a normal random variable is given by:

$$Prob[-z \leq X \leq z] = c$$

- With a symmetric distribution:

$$Prob[-z \leq X \leq z] = 1 - 2 \cdot Prob[X \geq z]$$

Confidence Limits

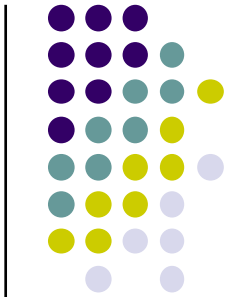


- Confidence limits for the normal distribution N($\mu = 0, \sigma = 1$)

Prob [X ≥ z]	0.1%	0.5%	1%	2.5%	5%	10%
z	3.09	2.58	2.33	1.96	1.65	1.28

- Thus: $Prob[-1.96 \leq X \leq +1.96] = 95\% \text{ (or } 0.95)$
  $Prob[-2.58 \leq X \leq +2.58] = 99\% \text{ (or } 0.99)$
- To use this we have to reduce our random variable *f* to have 0 mean and unit variance

Transform *f*



- Transformed value for *f* : (i.e. subtract the mean and divide by its *standard deviation*)

$$\frac{f-p}{\sqrt{\frac{p\cdot(1-p)}{N}}}$$

- Resulting equation:

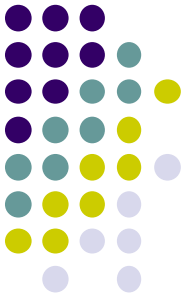
$$Prob[-z \leq \left(\frac{f-p}{\sqrt{\frac{p\cdot(1-p)}{N}}}\right) \leq +z] = c$$

- Solving for *p:*

$$p = \frac{\left(f + \frac{z^2}{2\cdot N} \pm z\cdot\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4\cdot N^2}}\right)}{1+\frac{z^2}{N}}$$

you have two values for *p* because you have a ± sign

Build Confidence Interval

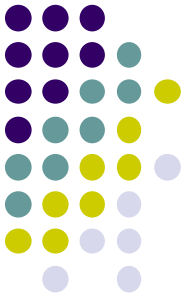


- A sequence of Bernoulli process follows a Normal distribution
- Resulting simpler equation

$$f - z_{\alpha/2} \cdot \sqrt{\frac{f \cdot (1-f)}{n}}$$

$$f + z_{\alpha/2} \cdot \sqrt{\frac{f \cdot (1-f)}{n}}$$

Examples



- f = 75%, N = **1,000**, c = 80% (so that z = 1.28):
  $p \in [0.732, 0.767]$
- f = 75%, N = **100**, c = 80% (so that z = 1.28):
  $p \in [0.695, 0.805]$

- Note that Normal distribution assumption is only valid for large N (i.e. $N > 100$)

- f = 75%, N = **10**, c = 80% (so that z = 1.28):
  $p \in [0.575, 0.925]$
  (should be taken with a grain of salt)

Better confidence intervals

- Expected success rate (estimation)

$$f' = (S+2) / (N+4)$$

for c = 95%

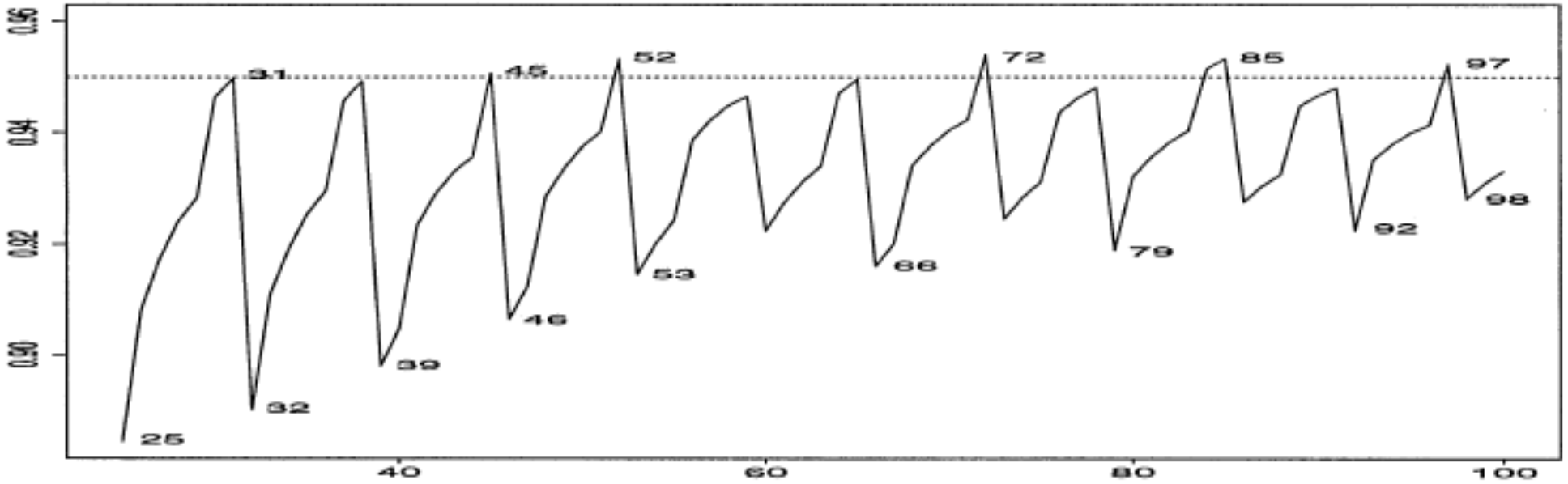
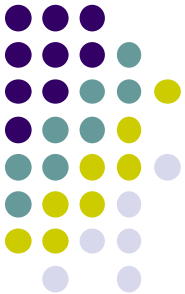


FIG. 1. *Standard interval; oscillation phenomenon for fixed* $p = 0.2$ *and variable* $n = 25$ *to* 100.

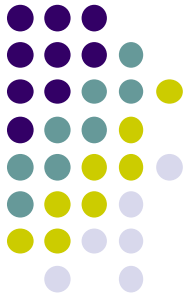
Brown, L.D., Cai, T.T, DasGupta, A. (2001). Interval estimation for a binomial proportion, *Statistical Science*, 16(2), 101-133

Holdout Estimation



- What to do if the amount of data is limited?
- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: one third for testing, the rest for training
- Problem: the samples might not be representative
  - Example: class might be missing in the test data
- Advanced version uses *stratification* (e.g. vote prediction)
  - Ensures that each class is represented with approximately equal proportions in both subsets
- Better solution?

Repeated Holdout Method

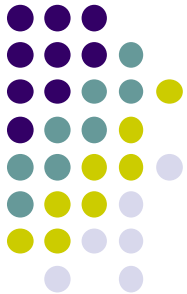


- Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
  - The error rates on the different iterations (err_i) are averaged to yield an overall error rate

$$\hat{err} = \frac{1}{k} \cdot \sum_{i=1}^{k} err_i$$

- This is called the *repeated holdout* method (or random subsampling)
- Still not optimum: the different test sets overlap
  - Can we prevent overlapping?

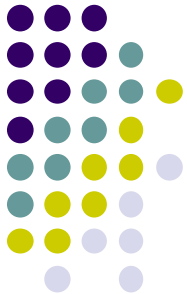
Repeated Holdout Method



- Comparison of the holdout and repeated holdout method

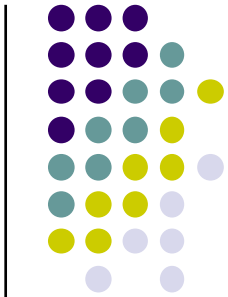
	Holdout	Random subsampling
Training instances	j	j
Testing instances	n-j	n-j
Iterations	1	k << n

Cross-Validation



- *Cross-validation* avoids overlapping test sets
  - First step: split data into *k* subsets of (roughly) equal size
  - Second step: use each subset in turn for testing, the remainder for training
- Called *k-fold cross-validation*

K-fold Cross-Validation

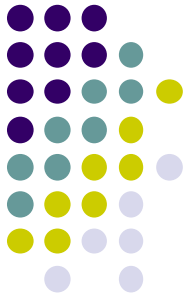


Example: 5-fold CV
4 parts for the training, one for the test
iterate five times

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5

err_1 err_2 err_3 err_4 err_5

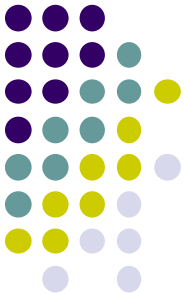
Cross-Validation



- *Cross-validation* avoids overlapping test sets
- Called *k-fold cross-validation*
- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

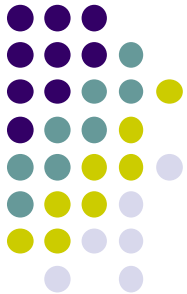
$$\hat{err} = \frac{1}{k} \cdot \sum_{i=1}^{k} err_i$$

Cross-Validation



- Standard method for evaluation:
  stratified ten-fold cross-validation
- Why ten?
  - Extensive experiments have shown that this is the best choice to get an accurate estimate
  - There is also some theoretical evidence for this
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Cross-Validation



Example accuracy rate

Same classifier (NSC)

n = 3,911

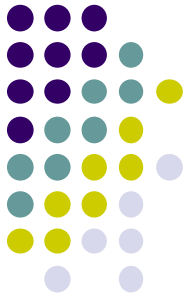
C = 20

Same data

Different folders

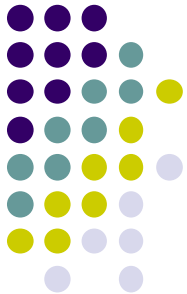
	Replication #1	Replication #2
1	0.8115	0.8046
2	0.8115	0.8276
3	0.8069	0.8138
4	0.8161	0.8207
5	0.7954	0.8046
6	0.8138	0.8299
7	0.8249	0.8180
8	0.8065	0.8226
9	0.8041	0.7696
19	0.8272	0.7742
mean	**0.8118**	**0.8085**

Leaving-One-Out



- Leaving-One-Out: a particular form of cross-validation:
  - Set number of folds to number of training instances
  - i.e., for *n* training instances, build classifier *n* times train on *n*-1 instances, evaluate on the *n*th
- Makes best use of the data
- Involves no random subsampling
- Very computationally expensive
  - (exception: NN)

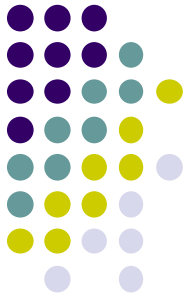
Leaving-One-Out & CV



- Comparison of the leaving-one-out and 10-fold CV

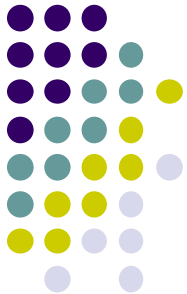
	Leave-one-out	10-fold CV
Training instances	n-1	90%
Testing instances	1	10%
Iterations	n	10

Leaving-One-Out



- Disadvantage of Leave-One-Out: stratification is not possible
  - It *guarantees* a nonstratified sample because there is only one instance in the test set!
- Time needed to compute / learn
- Extreme example: random dataset split equally into two classes
  - Best inducer predicts majority class
  - 50% accuracy on fresh data
  - Leave-One-Out-CV estimate is 100% error!
    (because if we have 50% in one case, and 50% in the other, the majority in the training set will always be in the wrong class)

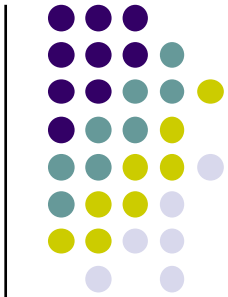
The Bootstrap



- CV uses sampling *without replacement*
  - The same instance, once selected, cannot be select again for a particular training/test set
- The general idea behind the *bootstrap* can be used to estimate the standard error of *various statistics* (e.g., median, correlation coef.)
- If we have a sample X = {x_1, x_2, …, x_i, …, x_n}, we can use it to generate *with replacement* other sample (with the same size *n*) and putting the probability 1/n on each observed value x_i (non-parametric bootstrap).

$$\hat{F} \rightarrow \mathbf{x}^* = (x_1^*, x_2^*, \cdots, x_n^*)$$

The Bootstrap



Procedure

1. Select B independent bootstrap samples $\mathbf{x}^{*1}, \mathbf{x}^{*2}, \cdots, \mathbf{x}^{*B}$ each consisting of *n* data values drawn *with replacement* from **X** (B is in the range 25 - 2,000)

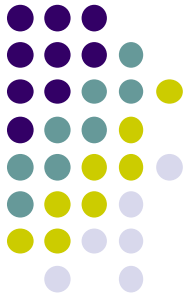
$$\hat{F} \rightarrow \mathbf{x}^{*i} = (x_1^*, x_2^*, \cdots, x_n^*)$$

2. Evaluate the bootstrap replication corresponding to each bootstrap sample

$$\hat{\theta}^*(b) = s(\mathbf{x}^{*b}) \text{ with } b = 1, 2, \cdots, B$$

where θ is the statistics of interest computed using the appropriate function s() (e.g., the median)

The Bootstrap



Procedure

3. Estimate the standard error $se_F(\hat{\theta})$
   by the sample standard deviation of the B replications

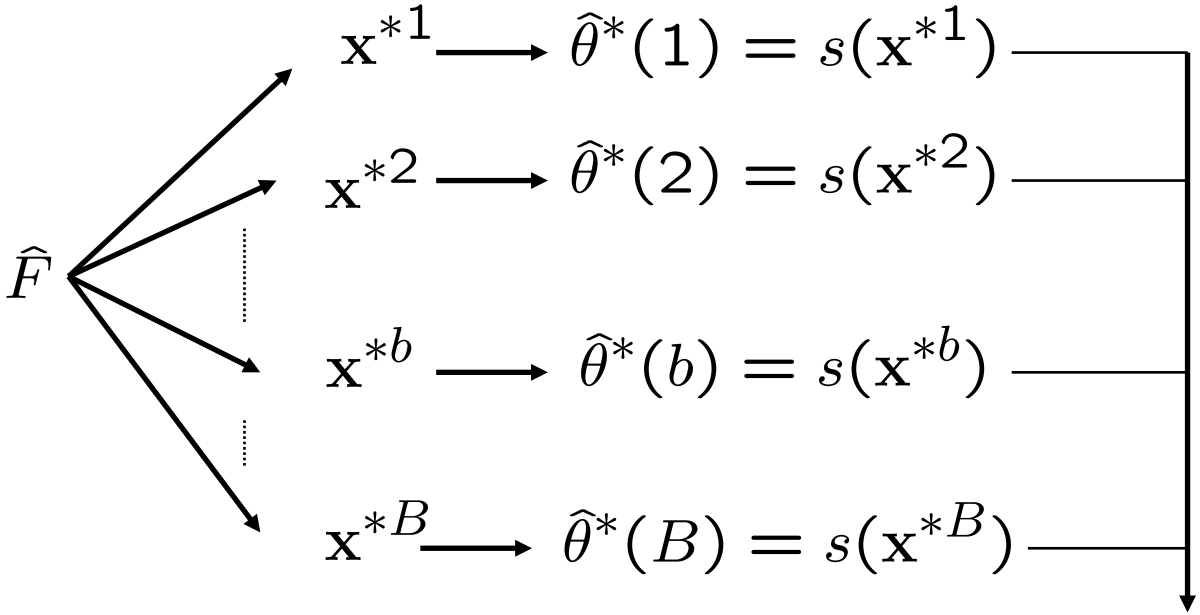
$$\hat{se}_B = \sqrt{\sum_{b=1}^{B} \left[\hat{\theta}^*(b) - \hat{\theta}^*(.)\right]^2 / (B-1)}$$

$$\text{with } \hat{\theta}^*(.) = 1/B \cdot \sum_{b=1}^{B} \hat{\theta}^*(b)$$

$$\text{and knowing that } \lim_{B \to \inf} \hat{se}_B = se_{\hat{F}} = se_{\hat{F}}(\hat{\theta}^*)$$

The Bootstrap

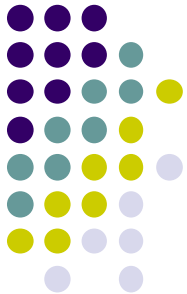
Example



$$\hat{se}_B = \sqrt{\sum_{b=1}^{B} \left[\hat{\theta}^*(b) - \hat{\theta}^*(.)\right]^2 / (B-1)}$$

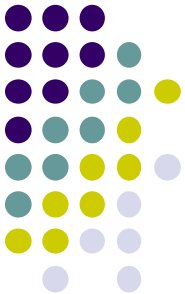
with $\hat{\theta}^*(.) = 1/B \cdot \sum_{b=1}^{B} \hat{\theta}^*(b)$

The Bootstrap



- In our context, the *bootstrap* uses sampling *with replacement* to form the training set
  - Sample a dataset of *n* instances *n* times *with replacement* to form a new dataset of *n* instances
  - Use this data as the training set
  - Use the instances from the original dataset that don't occur in the new training set for testing

The Bootstrap

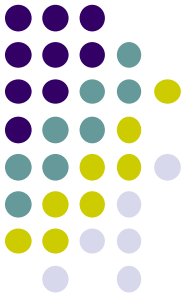


- Also called the *0.632 bootstrap*
- A particular instance has a probability 1/*n* of being picked
- Thus an instance has a probability of 1–(1/*n*) of *not* being picked
- Thus its probability of ending up in the test data is:

$$\left[1 - \frac{1}{n}\right]^n \approx e^{-1} \approx 0.368$$

- This means that the training data will contain approximately 63.2% of the instances

The Bootstrap

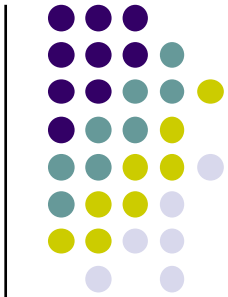


- The error estimate on the test data will be very pessimistic
  - Trained on just ~63% of the instances
- Therefore, combine it with the resubstitution error:

$$err = 0.632 \cdot e_{test\ instances} + 0.368 \cdot e_{training\ instances}$$

- The resubstitution error gets less weight than the error on the test data
- Repeat process several times with different replacement samples; average the results

The Bootstrap

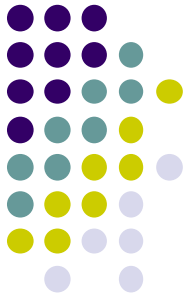


- Bootstrap estimators

	Bootstrap
Training instances	n (with j unique)
Testing instances	n-j
Iterations	200

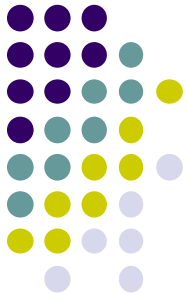
Efron b., & Tibshirani R.J.: An Introduction to the Bootstrap, Chapman & Hall, New York, 1993

Comparing Data Mining Schemes



- Frequent question: which of two learning schemes performs better?
- Note: this is domain dependent!
- Obvious way: compare 10-fold CV estimates
- Generally sufficient in applications
- However, what about machine learning research?
  - Need to show convincingly that a particular method works better

Comparing Data Mining Schemes



- Want to show that scheme A is better than scheme B in a particular domain
  - For a given amount of training data
  - On average, across all possible training sets
- Let's assume we have an infinite amount of data from the domain:
  - Sample infinitely many dataset of specified size
  - Obtain cross-validation estimate on each dataset for each scheme
  - Check if mean accuracy for scheme A is better than mean accuracy for scheme B

A Practical Example

Which one is the best?

Why?

n = 10

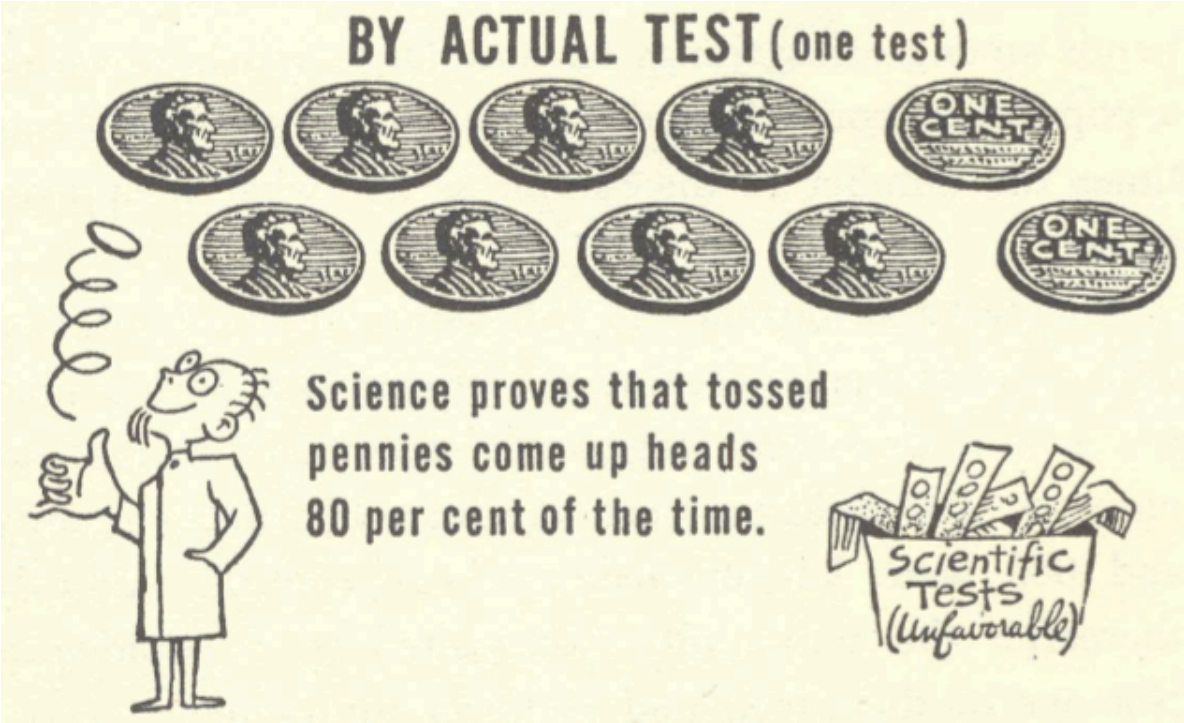
By intuition?

(toy-sided for demo only!)

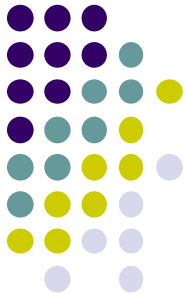
obs.	X_i	Y_i
1	0.84	0.88
2	0.78	0.97
3	0.67	0.74
4	0.87	0.80
5	0.80	0.87
6	0.78	0.90
7	0.78	0.90
8	0.79	0.86
9	0.82	0.84
10	0.81	0.78
sum	**7.94**	**8.54**

Formal (scientific) procedure

A scientific methodology



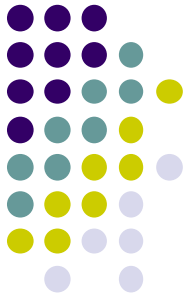
Paired *t*-Test



- In practice we have limited data and a limited number of estimates for computing the mean
- *Student's t-test* tells whether the means of two samples are significantly different
- In our case the samples are cross-validation estimates for different datasets from the domain
- Use a *paired t*-test because the individual samples are paired
- The same splitting for the two CV

William Gosset (1876 - 1937)
(working for Guiness)

Comparing Data Mining Schemes

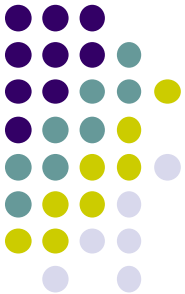


- If we have x_1, x_2, … x_k and y_1, y_2, … y_k are the $2k$ samples for the *k*-different datasets
- and the means $\bar{x}_x$ and $\bar{x}_y$
- With enough samples, the mean of a set of independent samples is normally distributed
- Estimated variances of the means are σ_x^2/k and σ_y^2/k
- If μ_x and μ_y are the true means then

$$\frac{\bar{x}_x - \mu_x}{\sigma_x/\sqrt{k}} \quad and \quad \frac{\bar{x}_y - \mu_y}{\sigma_y/\sqrt{k}} \quad \text{with} \quad \hat{\sigma}_x^2 = \frac{\sum_{i=1}^{k}(x_i - \bar{x}_x)^2}{k-1}$$

are *approximately* normally distributed with mean 0, standard deviation 1

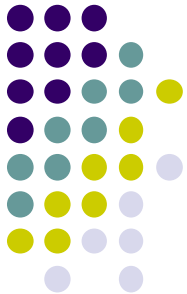
Student's Distribution



- With small samples ($k < 100$) the mean follows *Student's distribution with k–1 degrees of freedom*
- Confidence limits:

Prob [X ≥ lim]	*t* (9 dof)	*t* (10 dof)	*t* (20 dof)	Normal
0.05%	4.781	4.587	3.850	3.29
0.5%	3.250	3.169	2.845	2.58
1%	2.821	2.764	2.528	2.33
2.5%	2.262	2.228	2.086	1.96
5%	1.833	1.812	1.725	1.65
10%	1.383	1.3372	1.325	1.28

Student's Distribution

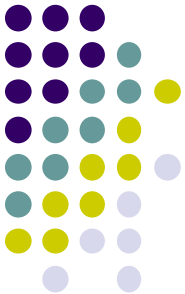


- Let $\bar{x}_d = \bar{x}_x - \bar{x}_y$ with $d_i = x_i - y_i$
- The difference of the means ($\bar{x}_d$) also has a Student's distribution with *k*–1 degrees of freedom
- Let σ_d^2 be the variance of the difference
- The standardized version of $\bar{x}_d$ is called the *t*-statistic:

$$t_{obs} = \frac{\bar{x}_d}{\hat{\sigma}_d/\sqrt{k}} \quad \text{with} \quad \hat{\sigma}_d = \sqrt{\frac{\sum_{i=1}^{k} d_i^2 - \frac{\left(\sum_{i=1}^{k} d_i\right)^2}{k}}{k-1}}$$

- We use t_{obs} to perform the *t*-test (*k*-1 dof)

Student's Distribution



- Fix a significance level
  - If a difference is significant at the α% level, there is a (100-α)% chance that the true means differ
- Divide the significance level by two because the test is two-tailed
  - i.e. the true difference can be +v or –v
- Look up the value for *z* that corresponds to $\alpha/2$
- If $t_{obs} \leq -z$ or $t_{obs} \geq z$ then the difference is significant
  - i.e. the *null hypothesis* (that the difference is zero) can be rejected

Student's Distribution

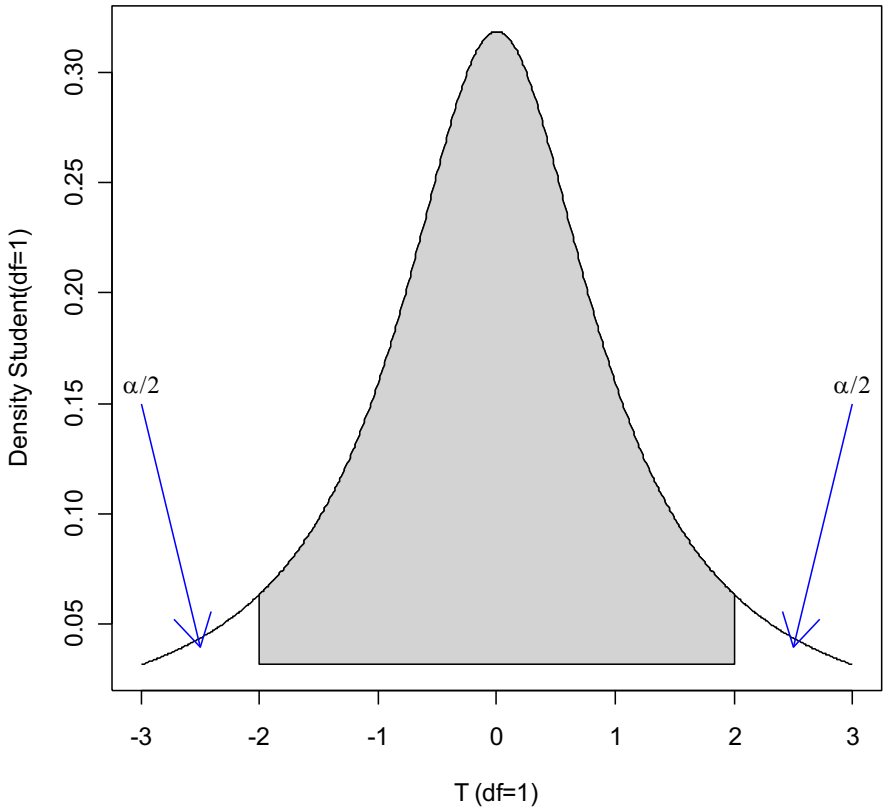
The Student distribution is symmetric.
Example with $\alpha = 5\%$

1) $\text{Prob}[-t_{lim} < T_{df=1} < t_{lim}] = 0.05$

2) $\text{Prob}[-t_{lim} < T_{df=7} < t_{lim}] = 0.05$

3) $\text{Prob}[-t_{lim} < T_{df=50} < t_{lim}] = 0.05$

$\rightarrow t_{lim\ df=1} = 12.71$
$\rightarrow t_{lim\ df=7} = 2.365$
$\rightarrow t_{lim\ df=50} = 2.009$



Example

n = 10
(or k = 10)

obs.	X_i	Y_i	X_i-Y_i	$(X_i-Y_i)^2$
1	0.84	0.88	-0.04	0.0016
2	0.78	0.97	-0.19	0.0361
3	0.67	0.74	-0.07	0.0049
4	0.87	0.80	0.07	0.0049
5	0.80	0.87	-0.07	0.0049
6	0.78	0.90	-0.12	0.0144
7	0.78	0.90	-0.12	0.0144
8	0.79	0.86	-0.07	0.0049
9	0.82	0.84	-0.02	0.0004
10	0.81	0.78	0.03	0.0009
sum	**7.94**	**8.54**	**-0.60**	**0.0874**

Example

When need to compute the value t_{obs} according to your data $t_{obs} = \frac{\bar{x}_d}{\sigma_d/\sqrt{k}}$

And then the estimated standard deviation

$$\hat{\sigma}_d = \sqrt{\frac{\sum_{i=1}^{k} d_i^2 - \frac{\left(\sum_{i=1}^{k} d_i\right)^2}{k}}{k-1}}$$

Example

$k = 10$

$$\hat{\sigma}_d = \sqrt{\frac{\sum_{i=1}^{k} d_i^2 - \frac{\left(\sum_{i=1}^{k} d_i\right)^2}{k}}{k-1}} \quad = 0.07557$$

$$t_{obs} = \frac{\bar{x}_d}{\sigma_d/\sqrt{k}} \quad = -0.06 / (0.07557/\sqrt{10}) = -2.51$$

Theoretical limit Student 9 dof, α = 5% t_{lim} = 2.262

Decision: Reject H_0

Another test: (unpaired) *t*-test t_{obs} = -2.231

Another View: Excel

A	B	C
1	X	Y
2	0.84	0.88
3	0.78	0.97
4	0.67	0.74
5	0.87	0.8
6	0.8	0.87
7	0.78	0.9
8	0.78	0.9
9	0.79	0.86
10	0.82	0.84
11	0.81	0.78

Data row

Two-tailed or on-tailed?

Paired=1 Normal=2

In cell E2

= TTEST(B2:B11, C2:C11, 2, 1)

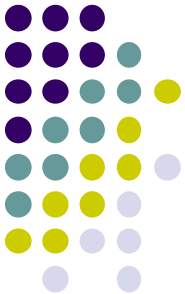
In cell E3

= TTEST(B2:B11, C2:C11, 2, 2)

A	D	E
1		
2	paired	3.33%
3	normal	3.86%
4		

Return the p-value, probability of having the corresponding t_{obs} value or larger

Predicting Probabilities

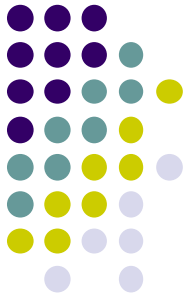


- Performance measure so far: success rate
- Also called *0-1 loss function*:

$$\sigma_i = \begin{cases} 0 & \text{if prediction is correct} \\ 1 & \text{if prediction is incorrect} \end{cases}$$

- Most classifiers produces class probabilities
- Depending on the application, we might want to check the accuracy of the probability estimates
- 0-1 loss is not the right thing to use in those cases

Quadratic Loss Function



- p_1, p_2, …, p_k are probability estimates for an instance (e.g., naïve Bayes)
- c is the index of the instance's actual class
- The actual class a_i
  a_1, …, a_k = 0, except for a_c which is 1
- *Quadratic loss* is:

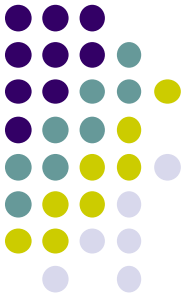
$$\sum_{j=1}^{k}(p_j - a_j)^2 = \sum_{j \neq c} p_j^2 + (a - p_c)^2$$

- We want to minimize

$$E\left[\sum_{j=1}^{k}(p_j - a_j)^2\right]$$

- Can show that this is minimized when $p_j = p^*_j$, the true probabilities

Informational Loss Function



- The informational loss function is

  $-\log_2(p_c)$

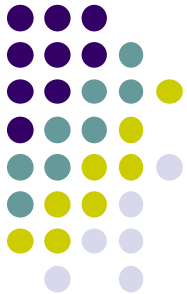
  where *c* is the index of the instance's actual class
- Number of bits required to communicate the actual class
- Let $p_1^*, p_2^*, \ldots, p_k^*$

  be the true class probabilities
- Then the expected value for the loss function is:

$$-p_1^* \cdot \log_2(p_1^*) - p_2^* \cdot \log_2(p_2^*) \ldots - p_k^* \cdot \log_2(p_k^*)$$

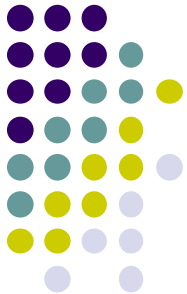
- Justification: minimized when $p_j = p_j^*$
- Difficulty: *zero-frequency problem*

Discussion



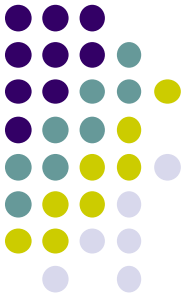
- Which loss function to choose?
  - Both encourage honesty
  - Quadratic loss function takes into account all class probability estimates for an instance
  - Informational loss focuses only on the probability estimate for the actual class
  - Quadratic loss is bounded: *it can never exceed 2*
  $$1 + \sum_{j=1}^{k} p_j$$
  - Informational loss can be infinite
- Informational loss is related to *MDL principle*

Counting the Cost



- In practice, different types of classification errors often incur different costs
- Examples:
  - Terrorist profiling
    “Not a terrorist” correct 99.99% of the time
  - Loan decisions
  - Oil-slick detection
  - Fault diagnosis
  - Promotional mailing

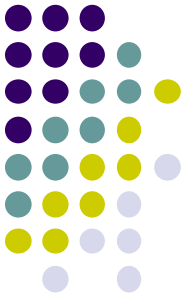
Counting the Cost



- The *confusion matrix*

		Predicted class	
		Yes (R+)	No (R-)
True state of nature	Yes (C+)	True Positive (TP)	False Negative (FN)
	No (C-)	False Positive (FP)	True Negative (TN)

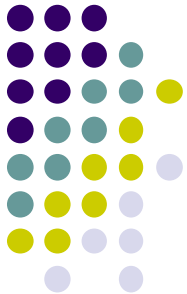
Counting the Cost



- The lie detector (US)
  "Do you tell the true?"

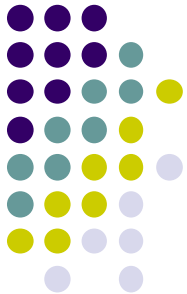
		Person	
		Truth (R+)	Lier (R-)
State of nature	True (C+)	True Positive (TP)	False Negative (FN)
	Lie (C-)	False Positive (FP)	True Negative (TN)

Counting the Cost



- Different measures can be used
- Accuracy = (TP+TN) / ((C+) + (C-))
- Sensitivity = TP / C+
  (the ability to correctly classify patients that actually have the disease, e.g. AIDS)
- Specificity = TN / C-
  (the ability to correctly classify patients that actually do not have the disease)
- Predictive value (+) = TP / R+
- Predictive value (-) = TN / R-
- There are many other types of cost!
  - e.g.: cost of collecting training data

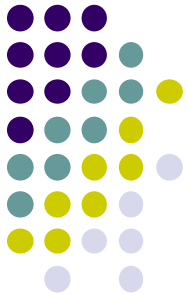
Precision, Recall, F_1, ...



- Percentage of detected examples as positive that are really positive: *Precision* = TP / (TP+FP)
- Percentage of positive examples that are detected as positive: *Recall* = TP / (TP+FN)
- Precision/recall curves have hyperbolic shape
  (If one increases, the other decreases)

- Trivial acceptor (always “yes”) (FN = TN = 0)
  Recall = TP / (TP + 0) = 100%
- Need both high recall and high precision!

Precision, Recall, F_1, ...

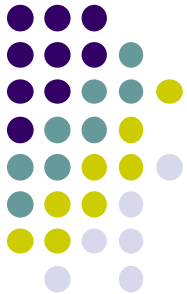


- Percentage of detected examples as positive that are really positive: *Precision* = TP / (TP+FP)
- Percentage of positive examples that are detected as positive: *Recall* = TP / (TP+FN)

- Mix the two measures as:
- Measure the performance using one single value, the F-measure
- *F-measure* =

  $F_1 = (2 \times \text{recall} \times \text{precision}) / (\text{recall} + \text{precision})$

Example



- Example: The lie detector (polygraph) can detect lies by considering biological measurements (blood pressure, respiration rhythm, heart frequency, …)
- Hypothesis: stress indicates a lie

- Performance: 83% to 89% lies are detected!
- But: 53% to 75% innocents correctly detected??
- Error: 12% to 47% (Vrij, 2008)

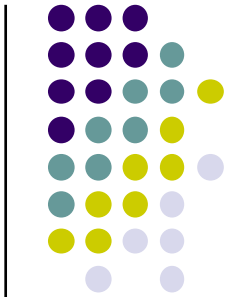
- Are you convince by the lie detector?

Example

- The lie detector (US)
  "Do you tell the true?"

		Person	
		Truth (R+)	Lier (R-)
State of nature	True (C+)	53% 75%	17% 11%
	Lie (C-)	47% 25%	83% 89%

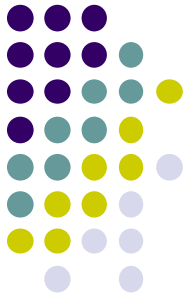
More than Two Classes



- When with have more than two classes

		Predicted class	
	Class *k*	Yes	No
True state of nature	Yes	True Positive (TP_k)	False Negative (FN_k)
	No	False Positive (FP_k)	True Negative (TN_k)

More than Two Classes



- Precision, Recall and F_1 measures must be given for each of the *m* classes
- For a single overall performance measure?
- Macro-averaging:
  one category = one vote

$$Prec = \sum_{k=1}^{m} \frac{Prec_k}{m}$$

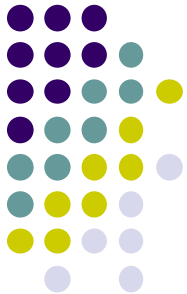
$$Recall = \sum_{k=1}^{m} \frac{Recall_k}{m}$$

- Micro-averaging:
  one decision = one vote

$$Prec = \frac{TP}{TP + FP} = \frac{\sum_{k=1}^{m} TP_k}{\sum_{k=1}^{m} (TP_k + FP_k)}$$

$$Recall = \frac{TP}{TP + FN} = \frac{\sum_{k=1}^{m} TP_k}{\sum_{k=1}^{m} (TP_k + FN_k)}$$

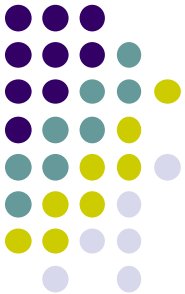
Example



Evaluation of a given learning scheme with four different datasets (macro-averaging)

Dataset	Precision	Recall	F_1
Iris	0.7	0.5	0.6
cars	0.5	0.6	0.55
TC 1	0.9	0.3	0.6
TC 2	0.7	0.5	0.6

Classification with Costs



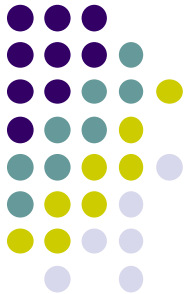
- Two cost matrices:

		Predicted class	
		yes	*no*
Actual class	*yes*	0	1
	no	1	0

		Predicted class		
		a	*b*	*c*
Actual class	*a*	0	1	1
	b	1	0	1
	c	1	1	0

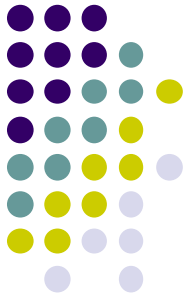
- Success rate is replaced by average cost per prediction
  - Cost is given by appropriate entry in the cost matrix

Classification with Costs



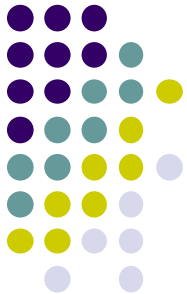
- Can take costs into account when making predictions
  - Basic idea: only predict high-cost class when very confident about prediction
- Given: predicted class probabilities
  - Normally we just predict the most likely class
  - Here, we should make the prediction that minimizes the expected cost
  - Expected cost: dot product of vector of class probabilities and appropriate column in cost matrix
  - Choose column (class) that minimizes expected cost

Cost-Sensitive Learning



- So far we haven't taken costs into account at training time
- Most learning schemes do not perform cost-sensitive learning
  - They generate the same classifier no matter what costs are assigned to the different classes
  - Example: standard decision tree learner
- Simple methods for cost-sensitive learning:
  - Resampling of instances according to costs
  - Weighting of instances according to costs
- Some schemes can take costs into account by varying a parameter, e.g. naïve Bayes

Evaluating Numeric Prediction

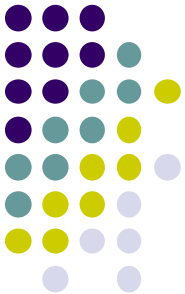


- Same strategies: independent test set, cross-validation, significance tests, etc.
- Difference: error measures
- Actual target values: a_1, a_2, …, a_n
- Predicted target values: p_1, p_2, …, p_n
- Most popular measure: *mean-squared error*

$$\frac{(p_1 - a_1)^2 + (p_2 - a_2)^2 + \ldots + (p_n - a_n)^2}{n}$$

- Easy to manipulate mathematically

Other Measures …



- The *root mean-squared error* :

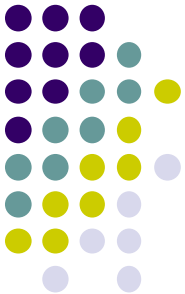
$$\sqrt{\frac{(p_1 - a_1)^2 + (p_2 - a_2)^2 + \ldots + (p_n - a_n)^2}{n}}$$

- The *mean absolute error* is less sensitive to outliers than the mean squared error:

$$\frac{|p_1 - a_1| + |p_2 - a_2| + \ldots + |p_n - a_n|}{n}$$

- Sometimes *relative* error values are more appropriate (e.g. 10% for an error of 50 when predicting 500)

Other Measures …



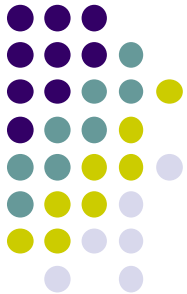
- How much does the scheme improve on simply predicting the average?
- The *relative squared error* is:

$$\frac{(p_1 - a_1)^2 + (p_2 - a_2)^2 + \ldots + (p_n - a_n)^2}{(\bar{a} - a_1)^2 + (\bar{a} - a_2)^2 + \ldots + (\bar{a} - a_n)^2}$$

- The *relative absolute error* is:

$$\frac{|p_1 - a_1| + |p_2 - a_2| + \ldots + |p_n - a_n|}{|\bar{a} - a_1| + |\bar{a} - a_2| + \ldots + |\bar{a} - a_n|}$$

Correlation Coefficient



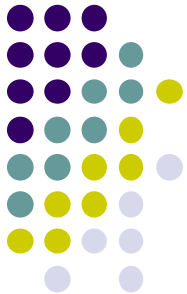
- Measures the *statistical correlation* between the predicted values and the actual values

$$\hat{\rho} = \frac{\hat{\sigma}_{pa}}{\sqrt{\hat{\sigma}_p \hat{\sigma}_a}} \qquad \hat{\sigma}_{pa} = \frac{\sum_{i=1}^{n}(p_i - \bar{p}) \cdot (a_i - \bar{a})}{n-1}$$

$$\hat{\sigma}_p = \frac{\sum_{i=1}^{n}(p_i - \bar{p})^2}{n-1} \quad \text{and} \quad \hat{\sigma}_a = \frac{\sum_{i=1}^{n}(a_i - \bar{a})^2}{n-1}$$

- Scale independent, between –1 and +1
- Good performance leads to large values!

Which Measure?

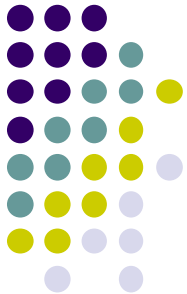


- Best to look at all of them
- Often it doesn't matter
- Example:

- D the best
- C the second-best

Error	A	B	C	D
Root mean-squared	67.8	91.7	63.3	57.4
Mean absolute	41.3	38.5	33.4	29.2
Root rel squared	42.2%	57.2%	39.4%	35.8%
Relative absolute	43.1%	40.1%	34.8%	30.4%
Correlation coefficient	0.88	0.88	0.89	0.91

The MDL Principle

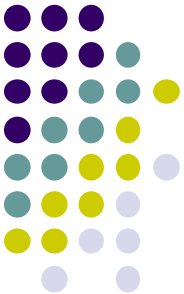


- MDL stands for *minimum description length*
- The description length is defined as:

  space required to describe a theory

  \+ *space required to describe the theory's mistakes*
- In our case the theory is the classifier and the mistakes are the errors on the training data
- Aim: we seek a classifier with minimal DL
- MDL principle is a *model selection criterion*

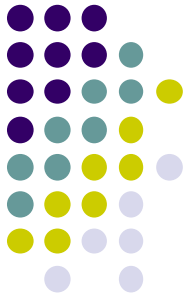
Model Selection Criteria



- Model selection criteria attempt to find a good compromise between:
  - The complexity of a model
  - Its prediction accuracy on the training data
- Reasoning: a good model is a simple model that achieves high accuracy on the given data
- Also known as *Occam's Razor* :
- the best theory is the smallest one that describes all the facts

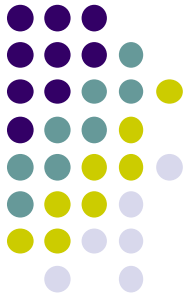
William of Ockham (about 1285-1348), the most influential philosopher of the 14th century

Elegance vs. Errors



- Theory 1: very simple, elegant theory that explains the data almost perfectly
- Theory 2: significantly more complex theory that reproduces the data without mistakes
- Theory 1 is probably preferable
- Classical example: Kepler's three laws on planetary motion

Conclusion



- Have two sets, a training and a test set
- Holdout, a first solution, 10-fold Cross-validation is better
- Give a confidence interval
- Use the *t*-test or better the paired *t*-test
- Take account of the error cost (if needed)
- Other measures: Precision, Recall, F_1
- Many measures for numerical prediction
- Take the complexity of the model into account (Occam's razor)