Course 395: Machine Learning - Lectures

• Lecture 1-2: Concept Learning (M. Pantic)

•Lecture 3-4: Decision Trees & CBC Intro (M. Pantic & S. Petridis)

➢ •Lecture 5-6: Evaluating Hypotheses (S. Petridis)

•Lecture 7-8: Artificial Neural Networks I (S. Petridis)

•Lecture 9-10: Artificial Neural Networks II (S. Petridis)

•Lecture 11-12: Instance Based Learning (M. Pantic)

•Lecture 13-14: Genetic Algorithms (M. Pantic)

Evaluating Hypotheses – Lecture Overview

- Measures of classification performance
  - Classification Error Rate
  - UAR
  - Recall, Precision, Confusion Matrix
  - Imbalanced Datasets
  - Overfitting
  - Cross-validation
- Estimating hypothesis accuracy
  - Sample Error vs. True Error
  - Confidence Intervals
  - Binomial and Normal Distributions
- Comparing Learning Algorithms
  - t-test

Classification Measures – Confusion Matrix

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

- Class 1: Positive
- Class 2: Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

- Visualisation of the performance of an algorithm
- Allows easy identification of confusion between between classes
  e.g. one class is commonly mislabelled as the other
- Most performance measures are computed from the confusion matrix

Classification Measures – Classification Rate

	Class 1 Predicted	**Class 2 Predicted**
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

- Class 1: Positive
- Class 2: Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

- Classification Rate / Accuracy: $$\frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}}$$
- Number of correctly classified examples divided by the total number of examples
- Classification Error = 1 – Classification Rate
- Classification Rate = Pr(correct classification)

Classification Measures – Recall

	Class 1 Predicted	**Class 2 Predicted**
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

- Class 1: Positive
- Class 2: Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

- Recall: $$\frac{TP}{TP + FN}$$

- Number of correctly classified positive examples divided by the total number of positive examples
- High recall: The class is correctly recognised (small number of FN)
- Recall = Pr(correctly classified | positive example)

Classification Measures – Precision

	Class 1 Predicted	**Class 2 Predicted**
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

- Class 1: Positive
- Class 2: Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

- Precision: $\dfrac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$
- Number of correctly classified positive examples divided by the total number of predicted positive examples
- High precision: An example labeled as positive is indeed positive (small number of FP)
- Precision = Pr(positive example | example is classified as positive)

Classification Measures – Recall/Precision

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

- Class 1: Positive
- Class 2: Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

- High recall, low precision: Most of the positive examples are correctly recognised (low FN) but there are a lot of false positives.

- Low recall, high precision: We miss a lot of positive examples (high FN) but those we predict as positive are indeed positive (low FP).

Classification Measures – F1 Measure/Score

- It is useful to have one number to measure the performance of the classifier
- $F_\alpha = (1+\alpha^2)\frac{Precision * Recall}{\alpha^2 * Precision + recall}$
- When α=1 → $F_1 = 2\frac{Precision * Recall}{Precision + recall}$

Classification Measures – UAR

	Class 1 Predicted	**Class 2 Predicted**
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

- Class 1: Positive
- Class 2: Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive
- TN: True Negative

- We compute recall for class1 (R1) and for class2 (R2).
- Unweighted Average Recall (UAR) = mean(R1, R2)

Classification Measures – Extension to Multiple Classes

	Class 1 Predicted	**Class 2 Predicted**	**Class 3 Predicted**
Class 1 Actual	TP	FN	FN
Class 2 Actual	FP	TN	?
Class 3 Actual	FP	?	TN

- In the multiclass case it is still very useful to compute the confusion matrix.
- We can define one class as positive and the other as negative.
- We can compute the performance measures in exactly the same way.

- CR = number of correctly classified examples (trace) divided by the total number of examples.
- Recall and precision and F1 are still computed for each class.
- UAR = mean(R1, R2, R3,…, RN)

Classification Measures – Balanced Dataset

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	70	30
Class 2 Actual	10	90

- CR: 80%
- Recall (cl.1): 70%
- Precision (cl.1): 87.5%
- F1 (cl.1): 77.8%
- UAR: 80%
- Recall (cl.2): 90%
- Precision (cl.2): 75%
- F1 (cl.2): 81.8%

- Balanced Dataset: The number of examples in each class are similar
- All measures result in similar performance

Classification Measures – Imbalanced Dataset Case 1: Both classifiers are good

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	700	300
Class 2 Actual	10	90

- CR: 71.8%
- Recall (cl.1): 70%
- Precision (cl.1): 98.6%
- F1 (cl.1): 81.9%
- UAR: 80%
- Recall (cl.2): 90%
- Precision (cl.2): 23.1%
- F1 (cl.2): 36.8%

- Imbalanced Dataset: Classes are not equally represented
- CR goes down, is affected a lot by the majority class
- Precision (and F1) for Class 2 is significantly affected –

  - 30% of class1 examples are misclassified→ leads to a higher number of FP than TN due to imbalance

Classification Measures – Imbalanced Dataset
Case 2: One classifier is useless

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	700	300
Class 2 Actual	100	0

- CR: 70%
- Recall (cl.1): 70%
- Precision (cl.1): 87.5%
- F1 (cl.1): 77.8%
- UAR: 35%
- Recall (cl.2): 0%
- Precision (cl.2): 0%
- F1 (cl.2): Not defined

- CR is misleading, one classifier is useless.
- F1 for class2 and UAR tell us that something is wrong.
- UAR also detects that there is a problem.

Classification Measures – Imbalanced Dataset Conclusions

- CR can be misleading, simply follows the performance of the majority class
- UAR is useful and can help to detect that one or more classifiers are not good but it does not give us any information about FP
- F1 is useful as well but is also affected by the class imbalance problem

  - We are not sure if the low score is due to one/more classifiers being useless or class imbalance

- That's why we should always have a look at the confusion matrix

Classification Measures – Imbalanced Dataset Some solutions

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	700	300
Class 2 Actual	10	90

Divide by the total number of examples per class $\longrightarrow$

	Class 1 Predicted	Class 2 Predicted
Class 1 Actual	0.7	0.3
Class 2 Actual	0.1	0.9

- Report performance ALSO on the "normalised matrix"

  - CR: 71.8%
  - Recall (cl.1): 70%
  - Precision (cl.1): 98.6%
  - F1 (cl.1): 81.9%
  - UAR: 80%
  - Recall (cl.2): 90%
  - Precision (cl.2): 23.1%
  - F1 (cl.2): 36.8%

$\longrightarrow$

  - CR: 80%
  - Recall (cl.1): 70%
  - Precision (cl.1): 87.5%
  - F1 (cl.1): 77.8%
  - UAR: 80%
  - Recall (cl.2): 90%
  - Precision (cl.2): 75%
  - F1 (cl.2): 81.8%

Classification Measures – Imbalanced Dataset Some solutions

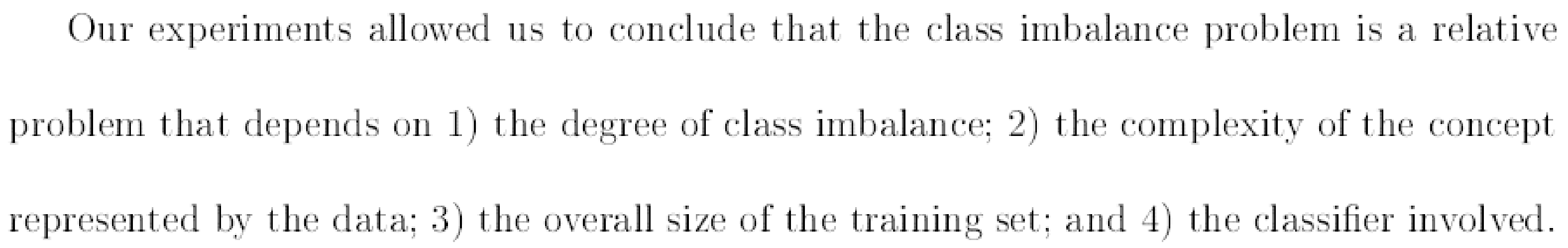
- Upsample the minority class
- Downsample the majority class

  - e.g. select randomly the same number of examples as the minority class.

  - Repeat this procedure several times and train a classifier each time with a different training set.

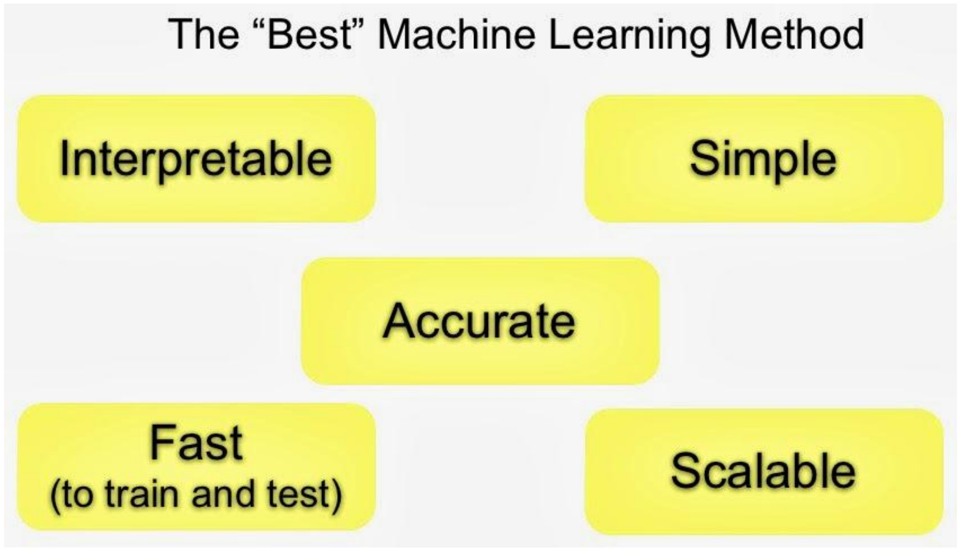
  - Report the mean and st. dev. of the selected performance measure

- Japkowicz, Nathalie, and Shaju Stephen. "The class imbalance problem: A systematic study." Intelligent data analysis 6.5 (2002): 429-449.

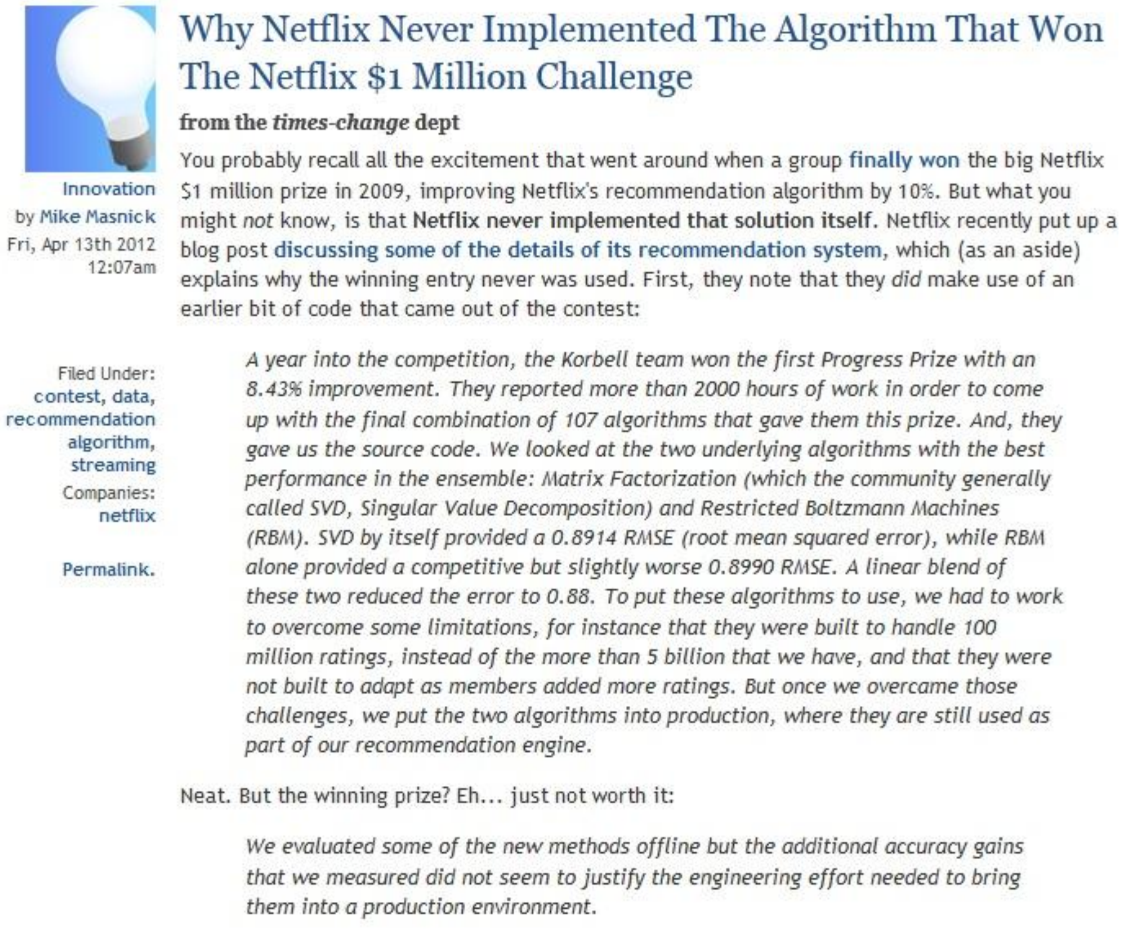


Our experiments allowed us to conclude that the class imbalance problem is a relative problem that depends on 1) the degree of class imbalance; 2) the complexity of the concept represented by the data; 3) the overall size of the training set; and 4) the classifier involved.

It’s not all about accuracy



http://radar.oreilly.com/2013/09/gaining-access-to-the-best-machine-learning-methods.html



Why Netflix Never Implemented The Algorithm That Won The Netflix $1 Million Challenge

from the *times-change* dept

Innovation
by Mike Masnick
Fri, Apr 13th 2012 12:07am

You probably recall all the excitement that went around when a group finally won the big Netflix $1 million prize in 2009, improving Netflix's recommendation algorithm by 10%. But what you might *not* know, is that **Netflix never implemented that solution itself.** Netflix recently put up a blog post discussing some of the details of its recommendation system, which (as an aside) explains why the winning entry never was used. First, they note that they *did* make use of an earlier bit of code that came out of the contest:

> *A year into the competition, the Korbell team won the first Progress Prize with an 8.43% improvement. They reported more than 2000 hours of work in order to come up with the final combination of 107 algorithms that gave them this prize. And, they gave us the source code. We looked at the two underlying algorithms with the best performance in the ensemble: Matrix Factorization (which the community generally called SVD, Singular Value Decomposition) and Restricted Boltzmann Machines (RBM). SVD by itself provided a 0.8914 RMSE (root mean squared error), while RBM alone provided a competitive but slightly worse 0.8990 RMSE. A linear blend of these two reduced the error to 0.88. To put these algorithms to use, we had to work to overcome some limitations, for instance that they were built to handle 100 million ratings, instead of the more than 5 billion that we have, and that they were not built to adapt as members added more ratings. But once we overcame those challenges, we put the two algorithms into production, where they are still used as part of our recommendation engine.*

Filed Under: contest, data, recommendation algorithm, streaming
Companies: netflix

Permalink.

Neat. But the winning prize? Eh... just not worth it:

> *We evaluated some of the new methods offline but the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring them into a production environment.*

https://www.techdirt.com/blog/innovation/articles/20120409/03412518422/

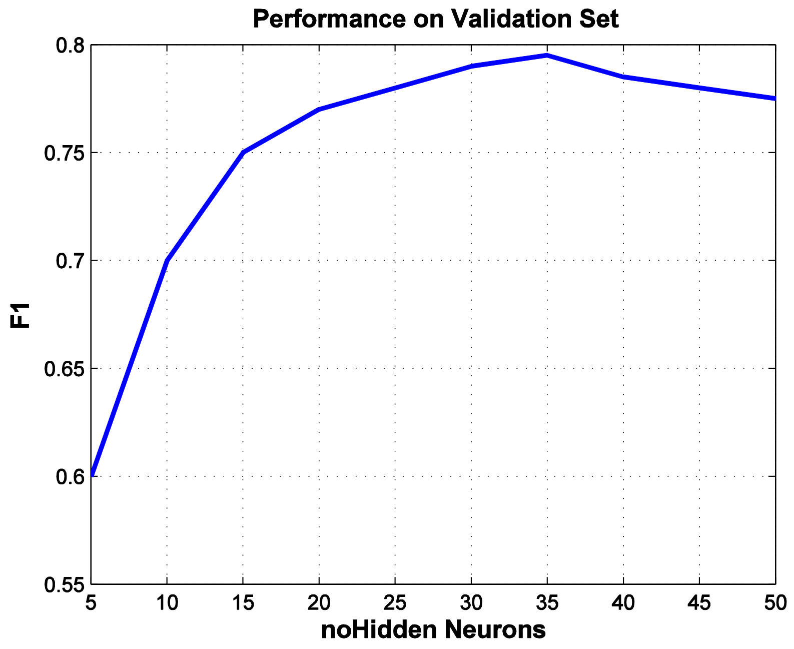
Training/Validation/Test Sets

- Split your dataset into 3 disjoint sets: Training, Validation, Test

- If a lot of data are available then you can try 50:25:25 otherwise 60:20:20.

- Identify which parameters need to be optimised and select a performance measure to evaluate the performance on the validation set.
  - e.g. number of hidden neurons
  - Use F1 as performance measure. It's perfectly fine to use any other measure, depends on your application

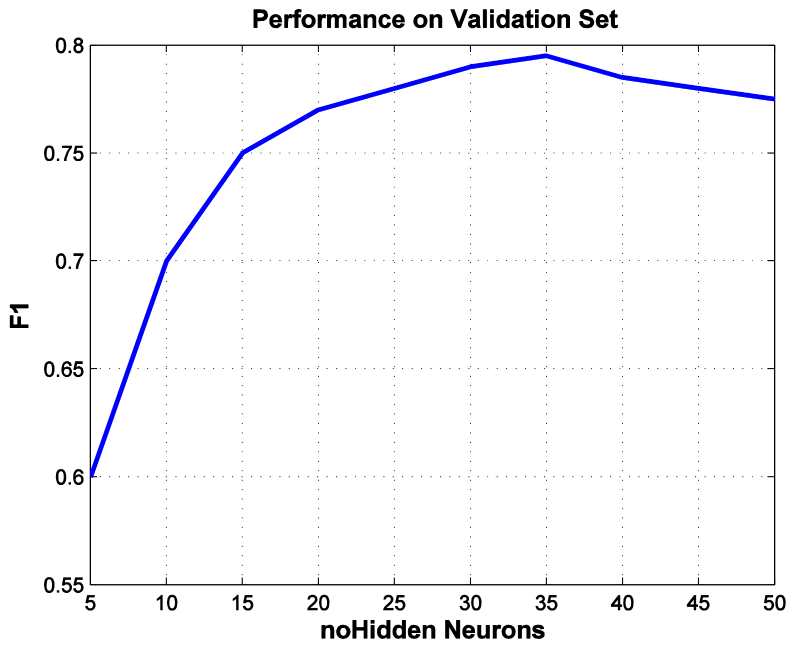
Training/Validation/Test Sets



- Train your algorithm on the training set multiple times, each time using different values for the parameters you wish to optimise.

- For each trained classifier evaluate the performance on the validation set (using the performance measure you have selected).

Training/Validation/Test Sets

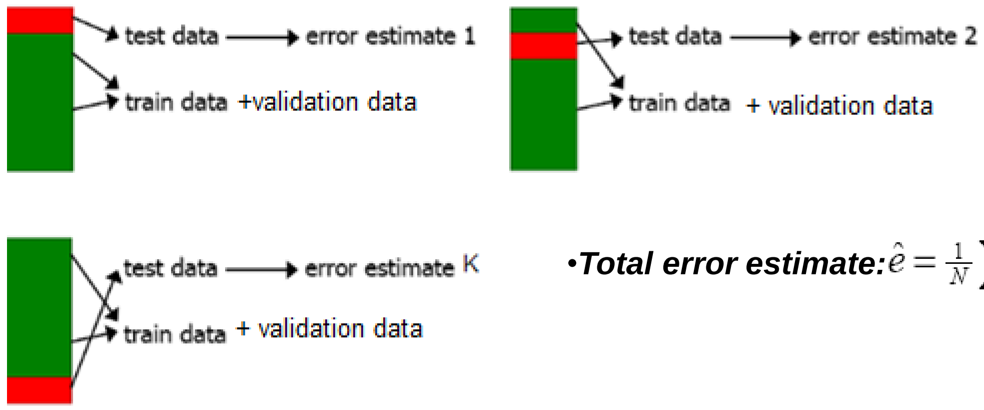


- Keep the classifier that leads to the maximum performance on the validation set (in this example the one trained with 35 hidden neurons)
- This is called parameter optimisation, since you select the set of parameters that have produced the best classifier.

Training/Validation/Test Sets

- Test the performance on the test set.
- The test set should **NOT** be used for training or validation. It is used **ONLY** in the end for estimating the performance on unknown examples, i.e. how well your trained classifiers generalises.
- You should assume that you do not know the labels of the test set and only after you have trained your classifier they are given to you.

Cross Validation

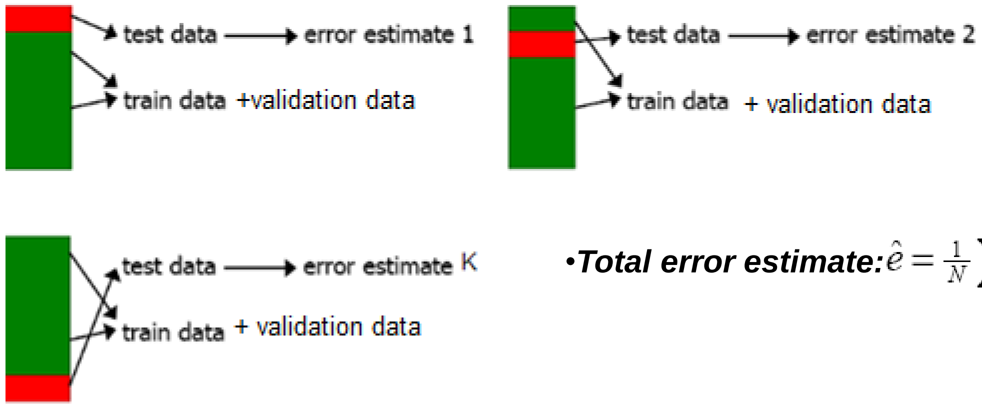

test data
error estimate 1
train data +validation data
test data
error estimate 2
train data + validation data
test data
error estimate K
train data + validation data

• ***Total error estimate:*** $\hat{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$

- When we have a lot of examples then the division into training/validation/test datasets is sufficient.

- When we have a small sample size then a good alternative is cross validation.

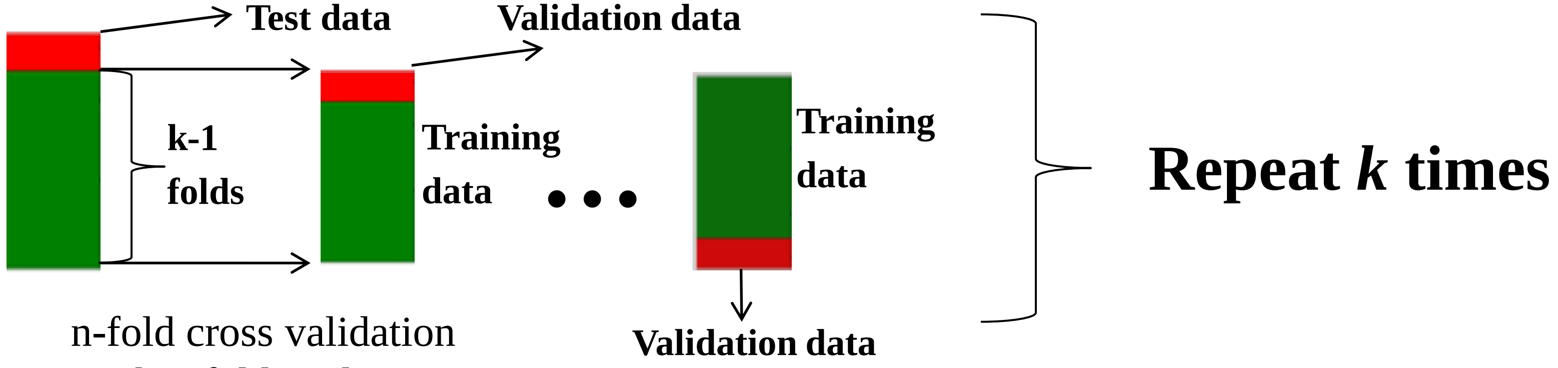
Cross Validation – Parameter Optimisation + Test Set Performance


test data
error estimate 1
train data +validation data
test data
error estimate 2
train data + validation data
test data
error estimate K
train data + validation data

•*Total error estimate:* $\hat{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$

- Divide dataset into *k* (usually 10) folds using *k*-1 for training+validation and one for testing
- Test data between different folds should never overlap!
- Training+Validation and test data in the same iteration should never overlap!
- In each iteration the error on the left-out test set is estimated
- Error estimate: average of the *k* errors

Cross Validation – Parameter Optimisation + Test Set Performance

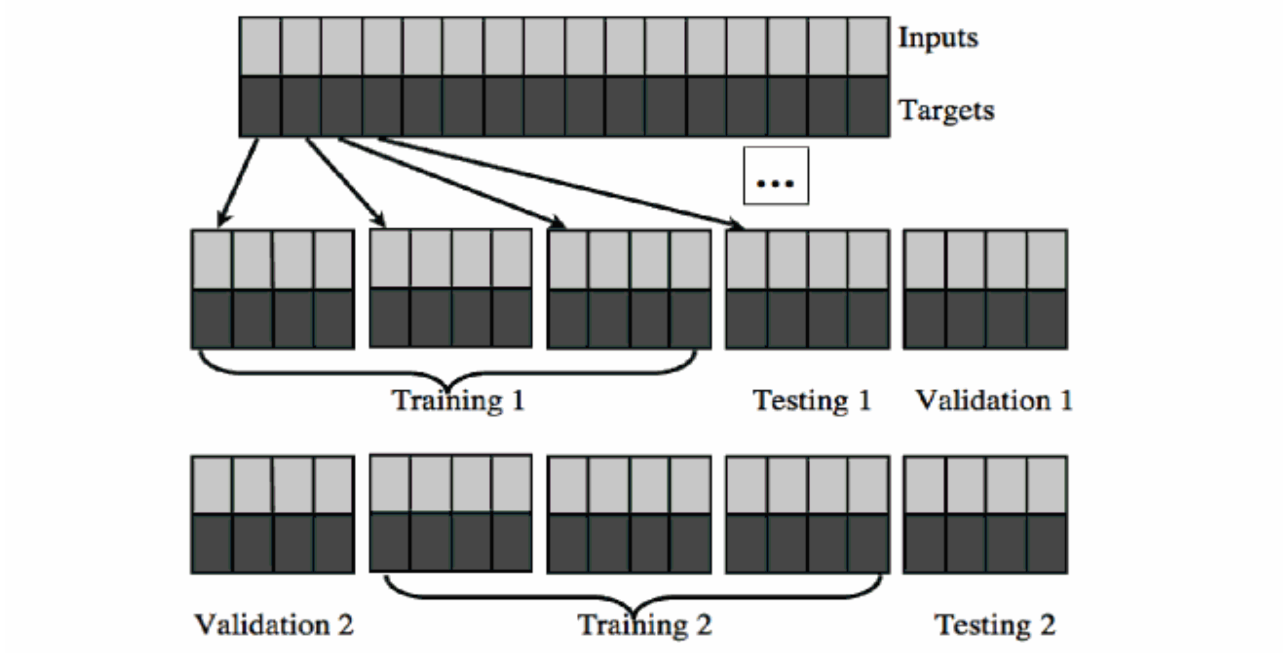


n-fold cross validation on k-1 folds only

- We can run an *n* (usually 2-3) fold cross-validation on the training+validation folds only in order to optimise the parameters.
- Select the parameters that result in the best average performance over all *n* folds.
- Then train on the entire training+validation set (*k*-1 folds) and test on the *k* fold.

- Inner cross-validation: Parameter Optimisation
- Outer cross-validation: Performance evaluation

Cross Validation – Parameter Optimisation + Test Set Performance

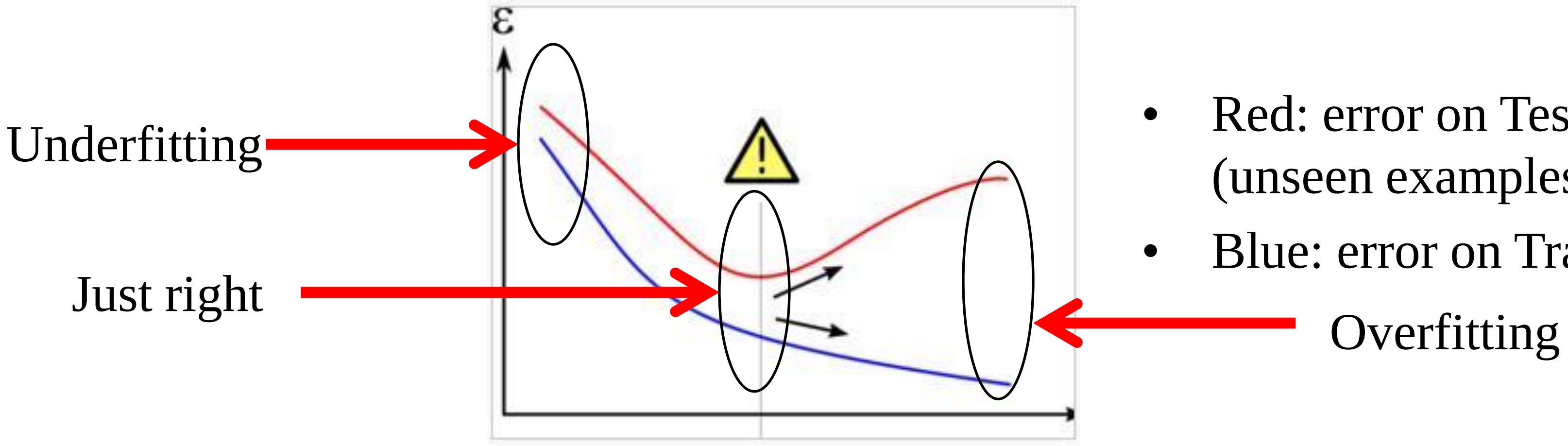


S. Marsland, Machine learning: An algorithmic perspective

- Another simpler way to optimise the parameters is simply to leave a second fold out for validation.
- Train on the training set, optimise parameters on the validation set and test on the test set.

Overfitting

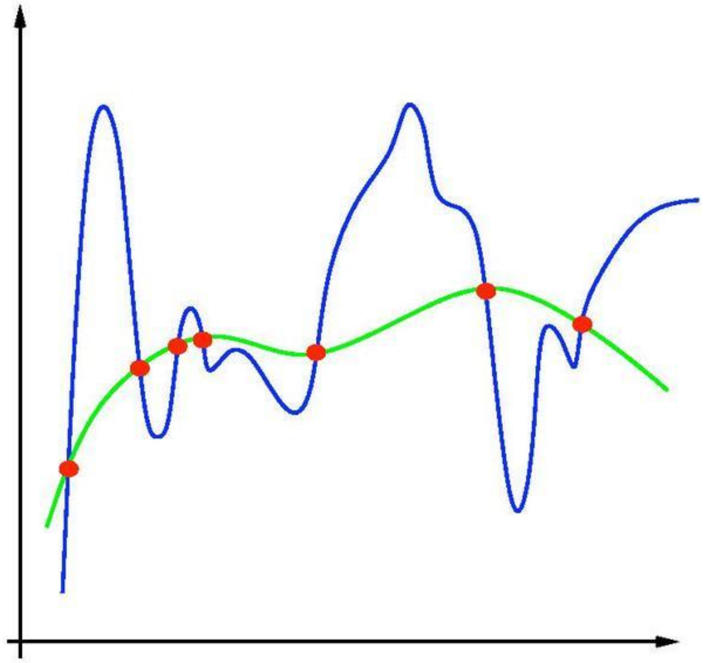
- *Given a hypothesis space H, h∈H overfits the training data if there exists some alternative hypothesis h′∈ H such that h has smaller error than h′ over the training examples, but h′ has smaller error than h over the entire distribution of instances.*



- Red: error on Test set (unseen examples)
- Blue: error on Training set

- Overfitting: Small error on training set, but large error on unseen examples.
- Underfitting: Larger error on training and test sets.

Overfitting



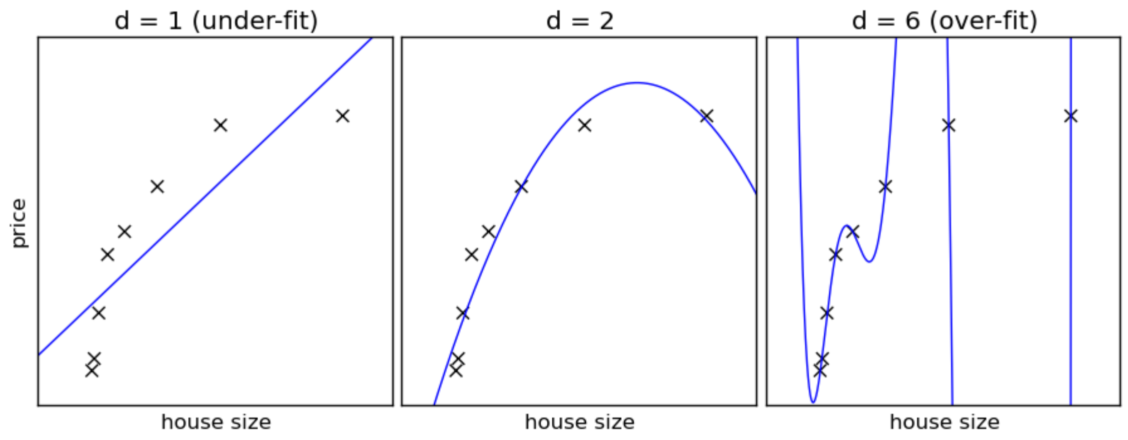
- Green: True target function
- Red: Training points
- Blue: What we have learned (overfitting)

(by Tomaso Poggio, http://www.mit.edu/~9.520/spring12/slides/class02/class02.pdf)

- The algorithm has learned perfectly the training examples, even the noise present in the examples and cannot generalise on unseen examples.

Overfitting

- Overfitting can occur when:
  - Learning is performed for too long (e.g. in Neural Networks).
  - The examples in the training set are not representative of all possible situations.
  - The model we use is too complex.



http://www.astroml.org/sklearn_tutorial/practical.html

Estimating accuracy of classification measures

- Q1: What is the best estimate of the accuracy over future examples drawn from the same distribution?

  - If future examples are drawn from a different distribution then we cannot generalise our conclusions based on the sample we already have.

- Q2: What is the probable error in this accuracy estimate? We want to assess the confidence that we can have in this classification measure.

Sample error & true error

- The True error of hypothesis *h* is the probability that it will misclassify a randomly drawn example *x* from distribution *D*:

$$error_D(h) \equiv \Pr[f(x) \neq h(x)]$$

***f*:true target function**

- The Sample error of hypothesis *h* based on a data sample S:

$$error_S(h) \equiv \frac{1}{n} \sum_{x \in S} \delta(f(x), h(x))$$

***n:* number of examples in S**

$\delta(f(x),h(x))=1$ if $f(x) \neq h(x)$

$\delta(f(x),h(x))=0$ if $f(x)=h(x)$

- We want to know the true error but we can only measure the sample error.

Sample Set Assumptions

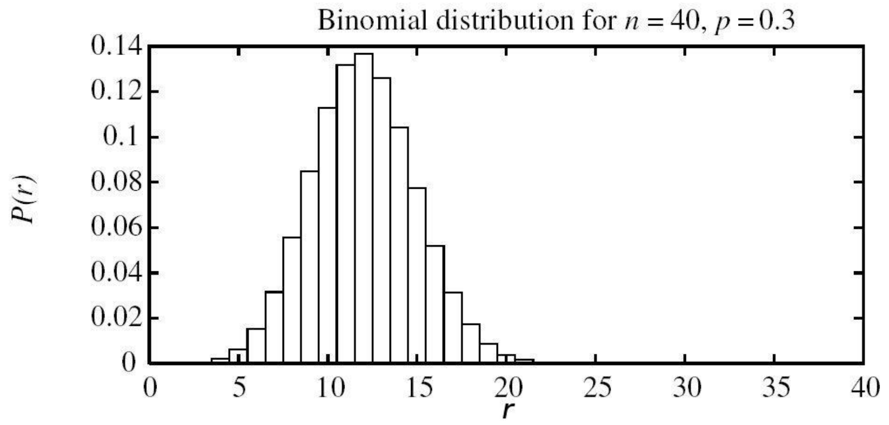
- We assume that the sample S is drawn at random using the same distribution D from which future examples will be drawn.
- Drawing an example from D does not influence the probability that another example will be drawn next.
- Examples are independent of the hypothesis (classifier) *h* being tested.

Bernouli Process

- Let's draw a random example from the distribution D (which generates our examples). This is a Bernouli trial since there are only two outcomes, the example will be either correctly classified or misclassified.
- The probability of misclassification is *p*. Note also that *p* is the true error.
- We draw *n* examples and count the number of misclassifications *r* (corresponds to the number of heads). Sample error = *r*/*n*.
- If we repeat the same experiment another *n* times then *r* will be slightly different.

Binomial Distribution

- If we plot the histogram of the sample error *r/n* then it will also look like the following plot:



- The number of errors (*r)* is a random variable that follows a Binomial distribution.
- The probability of observing *r* errors in a data sample of *n* randomly drawn examples is:

$$\Pr(R = r) = \frac{n!}{r!(n-r)!} p^r (1-p)^{n-r}$$

Sample Error as Estimator

- True error = p
- Sample error = r/n
- Sample error is a random variable that follows a binomial distribution.
- Estimator = random variable used to estimate some parameter (in our case p) of the population from which the sample is drawn.
- Sample error is called an estimator of the true error.
- Expected value of $r = np$ (Exp. Val. Binomial distribution)
- Expected value of sample error = $np/n = p$.

Sample Error as Estimator

- Q1: What is the best estimate of the accuracy over future examples drawn from the same distribution?
- True error = p
- Expected value of sample error = $np/n = p$.
- The best estimate of the true error is the sample error.

Confidence interval

- Q2: What is the probable error in this accuracy estimate? We want to assess the confidence that we can have in this classification measure.

- What we really want to estimate is a confidence interval for the true error.

- An *N*% confidence interval for some parameter *p* is an interval that is expected with probability *N%* to contain *p*.

  e.g. a 95% confidence interval [0.2,0.4] means that with probability 95% *p* lies between 0.2 and 0.4.

Trick

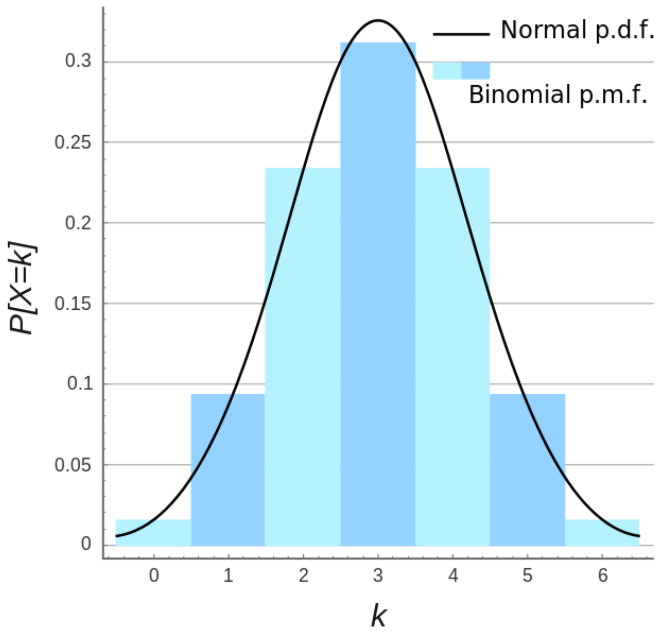
- For sufficiently large sample sizes the Binomial distribution can be approximated by a Normal (Gaussian) distribution.
- Sample size $n \geq 30$.
- So the sample error approximately follows a normal distribution with:

$$\mu = p \approx error_S$$

$$\sigma \approx \sqrt{\frac{error_S(1-error_S)}{n}}$$

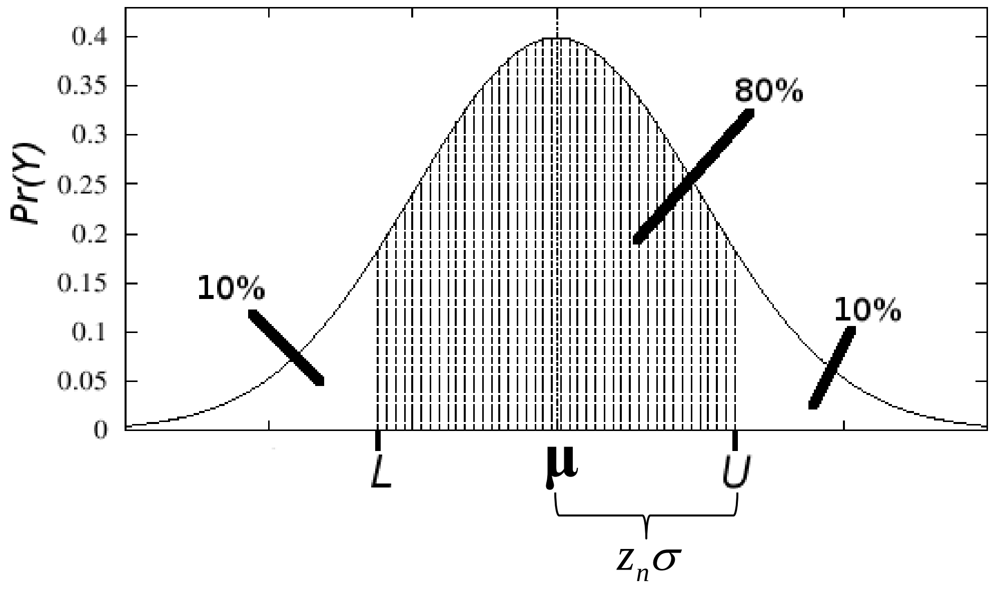
$\uparrow$

(p. 138 ML book)



by Xiao Fei

Confidence Interval



Normal distribution of sample error

$$\mu = p \approx error_S$$

$$\sigma \approx \sqrt{\frac{error_S(1-error_S)}{n}}$$

- The probability that the sample error will fall between L and U is $\int_L^U \Pr(Y)$ for this example it is 80%.
- In other words, the sample error will fall between $[\mu - z_n\sigma, \mu + z_n\sigma]$ N% of the time (in this example 80%) .
- Similarly, we can say that μ will fall between $[error_S - z_n\sigma, error_S + z_n\sigma]$ N% of the time.

Confidence interval - Theory

Given a sample S with $n >= 30$ on which hypothesis h makes r errors, we can say that:

Q1: The most probable value of $error_D(h)$ is $error_s(h)$

Q2: With N % confidence, the true error lies in the interval:

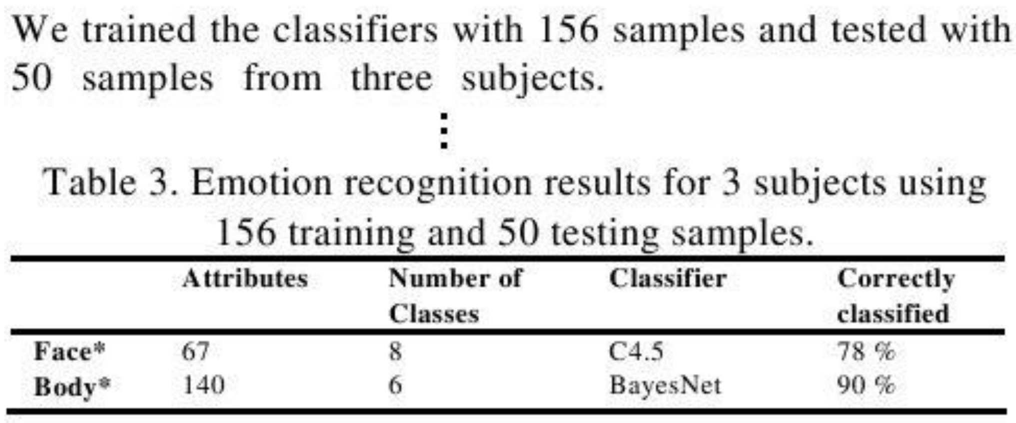
$$error_s(h) \pm z_N \sqrt{\frac{error_s(h)(1-error_s(h))}{n}}$$

with:

$N\%$:	50%	68%	80%	90%	95%	98%	99%
z_N:	0.67	1.00	1.28	1.64	1.96	2.33	2.58

Confidence interval – example (2)

Given the following extract from a scientific paper on multimodal emotion recognition:



We trained the classifiers with 156 samples and tested with 50 samples from three subjects.

⋮

Table 3. Emotion recognition results for 3 subjects using 156 training and 50 testing samples.

	Attributes	Number of Classes	Classifier	Correctly classified
Face*	67	8	C4.5	78 %
Body*	140	6	BayesNet	90 %

For the Face modality, what is n? What is $error_s(h)$?

Exercise: compute the 95% confidence interval for this error.

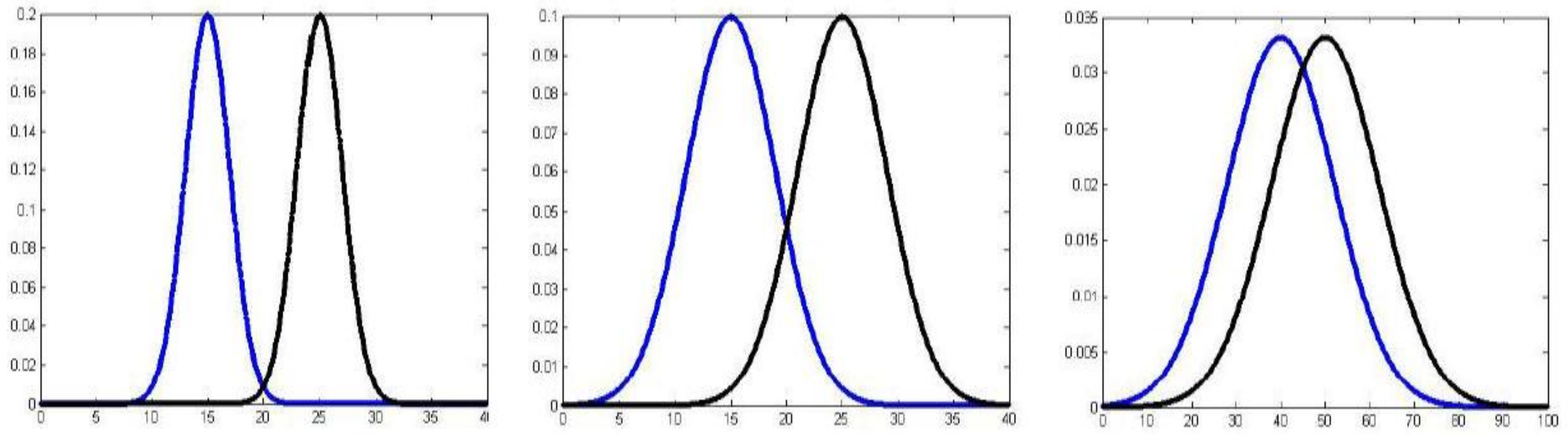
Confidence interval – example (3)

Given that *error*$_s$(*h*)=0.22 and n= 50, and z_N=1.96 for N = 95%, we can now say that with 95% confidence *error*$_D$(*h*) will lie in the interval:

$$\left[0.22-1.96\sqrt{\frac{0.22(1-0.22)}{50}},0.22+1.96\sqrt{\frac{0.22(1-0.22)}{50}}\right]=$$
$$[0.11,0.34]$$

What will happen when $n \to \infty$?

Comparing Two Algorithms



- Consider the distributions as the classification errors of two different classifiers derived by cross-validation.
- The means of the distributions are not enough to say that one of the classifiers is better!! In all cases the mean difference is the same.
- That’s why we need to run a statistical test to tell us if there is indeed a difference between the two distributions.

Two-sample T-test

- Null hypothesis: two sets of observations **x, y** are independent random samples from normal distributions with equal means.

- For example **x, y** could be the classification errors on two different datasets.

- We define the test statistic as: $t = \dfrac{\bar{\mu}_x - \bar{\mu}_y}{\sqrt{\dfrac{\sigma_x^2}{n} + \dfrac{\sigma_y^2}{m}}}$

- μ_x, μ_y *are the sample means*
- σ_x^2, σ_y^2 are the sample variances
- n, m are the sample sizes

Paired T-test

- Null hypothesis: the difference between the observations **x-y** are a random sample from a normal distribution with $\mu = 0$ and unknown variance.
- It's called paired because the observations are matched, they are not independent.

- For example **x, y** could be the classification errors on the same folds of cross-validation from two different algorithms. The test folds are the same, i.e. they are matched.

- We define the test statistic as: $t = \dfrac{\bar{\mu}_{x-y}}{\sqrt{\dfrac{\sigma^2_{x-y}}{n}}}$

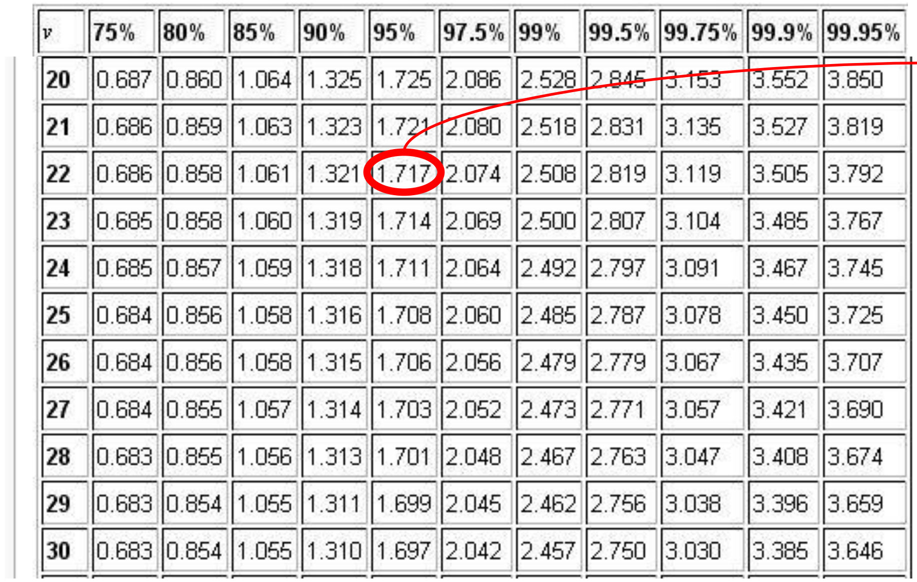
- μ_{x-y} is the sample mean of the differences
- $\sigma^2_{x-y}$is the sample variance of the differences.
- n is the sample size

T-test

- The test statistic *t* will follow a *t*-distribution if the null hypothesis is true. That is why it is called t-test.
- Once we compute the test statistic we also define a confidence level, usually 95%.

Confidence Level

Degrees of freedom: number of values that are free to vary, e.g. for paired t-test = *n*-1.



v	75%	80%	85%	90%	95%	97.5%	99%	99.5%	99.75%	99.9%	99.95%
20	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.153	3.552	3.850
21	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.135	3.527	3.819
22	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.119	3.505	3.792
23	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.104	3.485	3.767
24	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.091	3.467	3.745
25	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.078	3.450	3.725
26	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.067	3.435	3.707
27	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.057	3.421	3.690
28	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.047	3.408	3.674
29	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.038	3.396	3.659
30	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.030	3.385	3.646

t is less than 1.717 with probability 95%.

T-test

- If the calculated *t* value is above the threshold chosen for statistical significance then the null hypothesis that the two groups do not differ is rejected in favour of the alternative hypothesis, which typically states that the groups do differ.

- Significance level = 1 – confidence level, so usually 5%.

- Significance level α*%:* α times out of 100 you would find a statistically significant difference between the distributions even if there was none. It essentially defines our tolerance level.

- To summarise: we only have to compute *t*, set α and we use a lookup table to check if our value *t* is higher than the value in the table. If yes, then our sets of observations are different (null hypothesis rejected).