Course 395: Machine Learning - Lectures

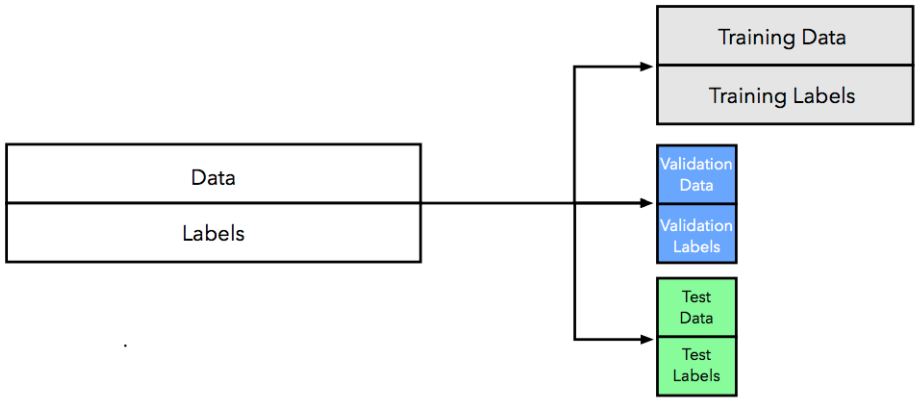
- Lecture 1-2: Concept Learning (M. Pantic)
- Lecture 3-4: Decision Trees & CBC Intro (M. Pantic & S. Petridis)
- Lecture 5-6: Evaluating Hypotheses (S. Petridis)
- Lecture 7-8: Neural Networks I (S. Petridis)
- Lecture 9-10: Neural Networks II (S. Petridis)
- Lecture 11-12: Neural Networks III (S. Petridis)
- Lecture 13-14: Genetic Algorithms (M. Pantic)

Evaluating Hypotheses – Lecture Overview

- Training / Parameter Optimisation / Evaluation
  - Holdout Method
  - Cross-validation
- Measures of classification performance
  - Confusion Matrix
  - Classification Error/Rate
  - Unweighted Average Recall (UAR)
  - Recall, Precision, F1 measure
  - Imbalanced Datasets
  - Overfitting
- Estimating hypothesis accuracy
  - Sample Error vs. True Error
  - Confidence Intervals
- Comparing Learning Algorithms
  - t-test

Holdout Method

- 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.



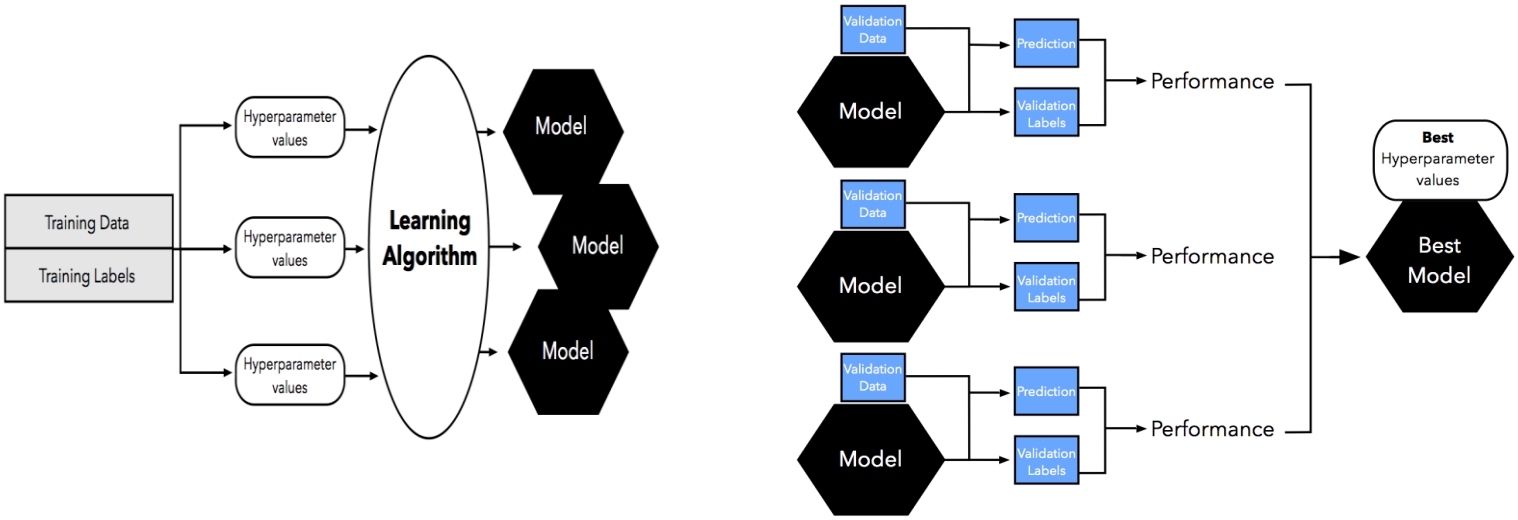
From: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html

Holdout Method

- Identify which parameters need to be optimised
  - e.g. number of hidden neurons, number of hidden layers etc

- Select a performance measure to evaluate the performance on the validation set
  - F1, Classification Rate etc
  - Appropriate measure depends on the application, if the test set is imbalanced etc

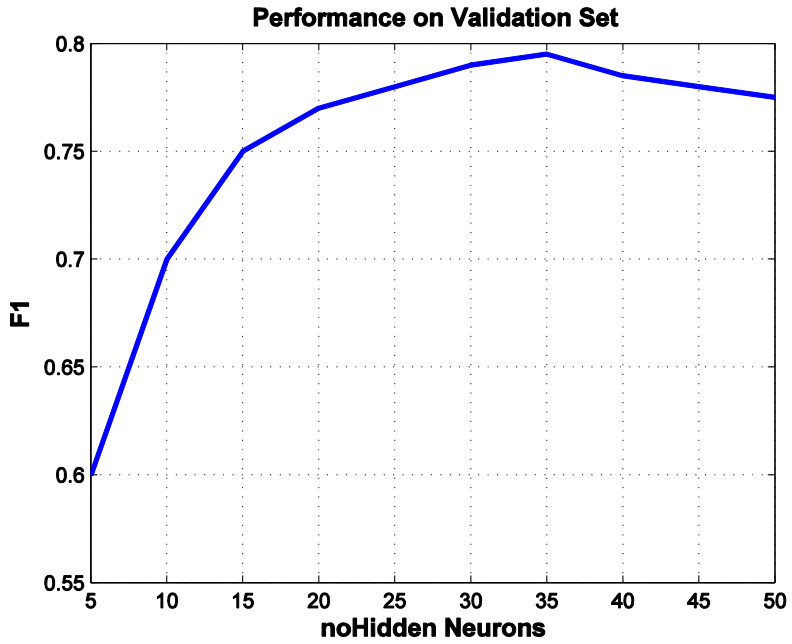
Holdout Method



From: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html

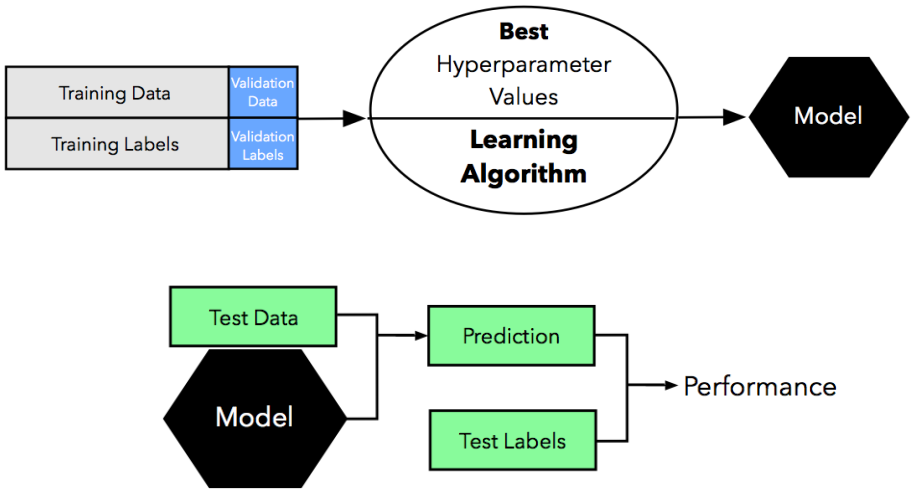
- Train your algorithm on the training set multiple times, each time using different values for the parameters you wish to optimise.
- For each trained classifier evaluate the performance on the validation set (using the performance measure you have selected).

Holdout Method



- 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 optimization/tuning, since you select the set of parameters that have produced the best classifier.

Holdout Method

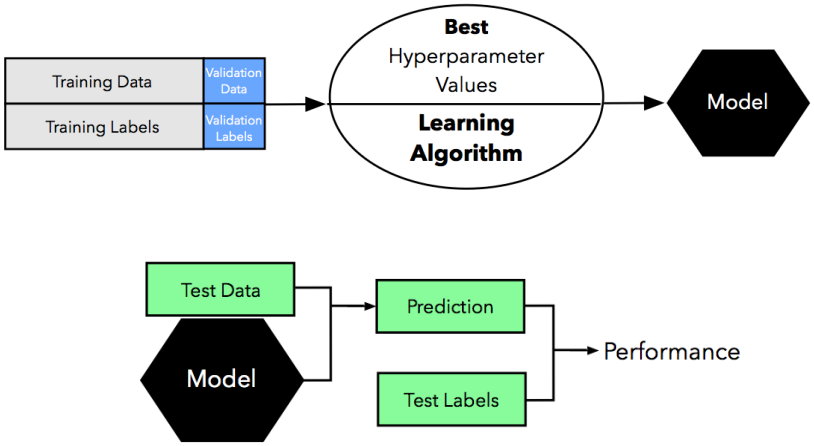


From: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html

- You can either merge the training and validation sets and train a new classifier using the optimal set of parameters OR you can simply use the best classifier (trained only on the training set).

- Test the performance on the test set.

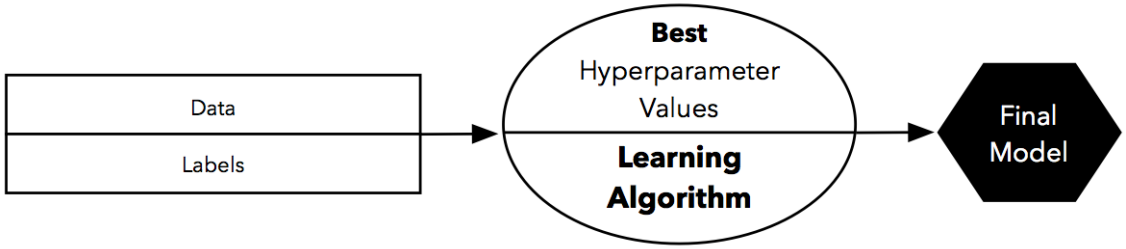
Holdout Method



From: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html

- 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 classifier 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.

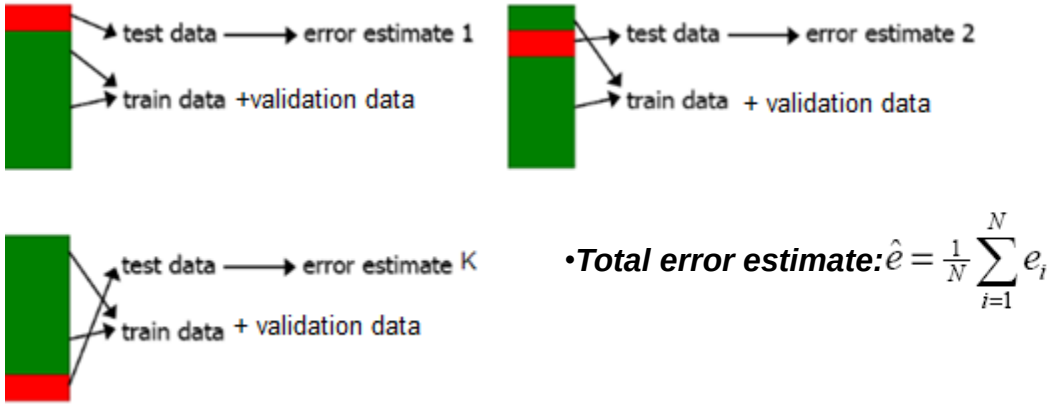
Holdout Method



From: https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html

- We need a model which we will use for classifying new examples.
- Either use the one trained on the training set or on training + validation sets OR train a new model on the entire dataset using the optimal set of parameters.

Cross Validation

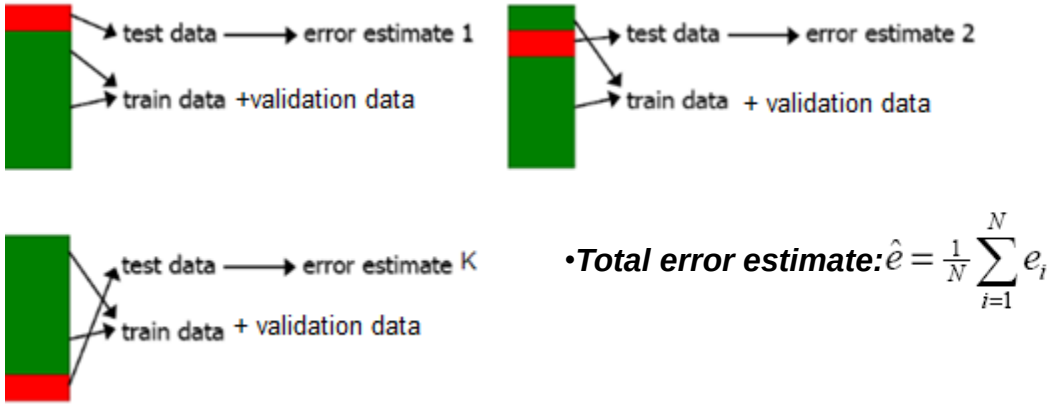


•***Total error estimate:*** $\hat{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$

- 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 – Test Set Performance Estimation

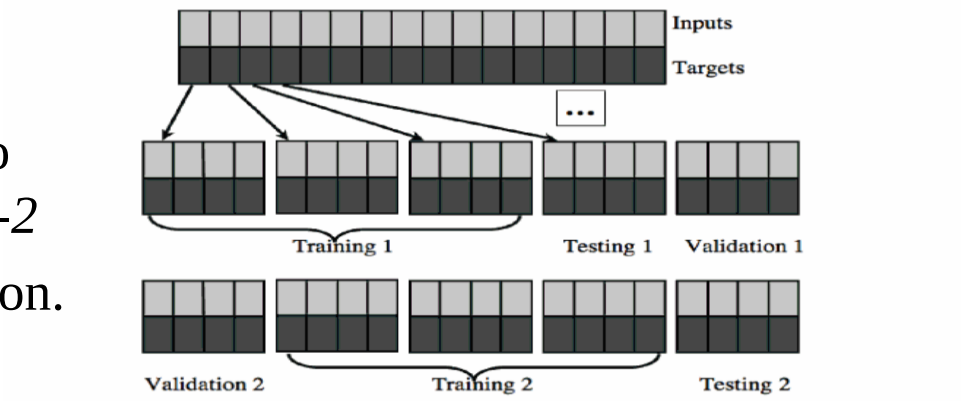


•***Total error estimate:*** $\hat{e} = \frac{1}{N}\sum_{i=1}^{N} e_i$

- 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– Test Set Performance Estimation

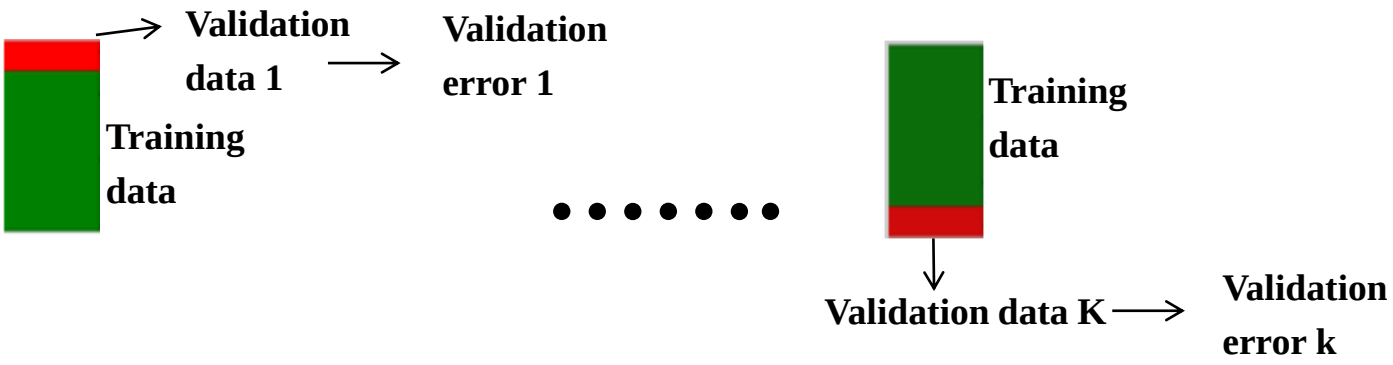
- The *k-1* folds should be divided into training and validation folds, e.g. *k-2* folds for training and 1 for validation.



S. Marsland, Machine learning: An algorithmic perspective

- Train on the training set, optimise parameters on the validation set and test on the test set.
- We can only estimate the test set performance. In other word we evaluate how our implementation (and the way we optimise the parameters) generalises on unknown test sets.
- We know nothing about the optimal set of parameters. We find a different set of optimal parameters in each fold.

Cross Validation – Parameter Estimation



- We can use cross validation to estimate the optimal set of parameters
- *k-1*folds for training, 1 fold left out for validation (using the entire dataset)
- For each parameter set run the *k* fold cross-validation
- Select the parameters that result in the best average performance over all *k* left out folds

Parameter Optimisation–Performance Estimation - Summary

- **CASE 1:** A lot of data are available (Holdout Method)

  1) Tune parameters on validation set

  2) Estimate generalization performance using the test set

  3) Train on entire dataset using optimal set of parameters

- **CASE 2:** Data are limited (Cross validation)

  1) Run cross validation to estimate the test set performance

  - Training, validation, test folds

  - Optimise parameters in each iteration

  2) Run cross validation to estimate optimal parameters

  - Training, Validation folds only

  3) Train on entire dataset using optimal set of parameters

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 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: $$\frac{TP + TN}{TP + TN + FP + FN}$$

- 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: $\frac{TP}{TP + FP}$
- 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

Recall: $\frac{TP}{TP + FN}$ Precision: $\frac{TP}{TP + FP}$

- 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 others 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 Test Set

	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 (of the test set) are similar
- All measures result in similar performance

Classification Measures – Imbalanced Test Set

Case 1: Both classes are classified with same recall as before

	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 are significantly affected –

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

Classification Measures – Imbalanced Test Set
Case 2: One class is completely misclassified

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

- CR: 63.6%
- 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, class 2 is completely misclassified.
- F1 for class 2 shows that something is wrong.
- UAR also detects that there is a problem.

Classification Measures – Imbalanced Test Set Conclusions

- CR can be misleading, simply follows the performance of the majority class
- UAR is useful and can help to detect that one class is completely misclassified 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 class being misclassified or class imbalance

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

Classification Measures – Imbalanced Test Set 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 →

	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%

→

- 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 Test Set 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 →

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

- These would be the results if we had the same number of examples and the performance of the classifier remained the same

- We don't have the same number of examples and there is no guarantee that the performance will remain the same (but still it's one solution to the problem)

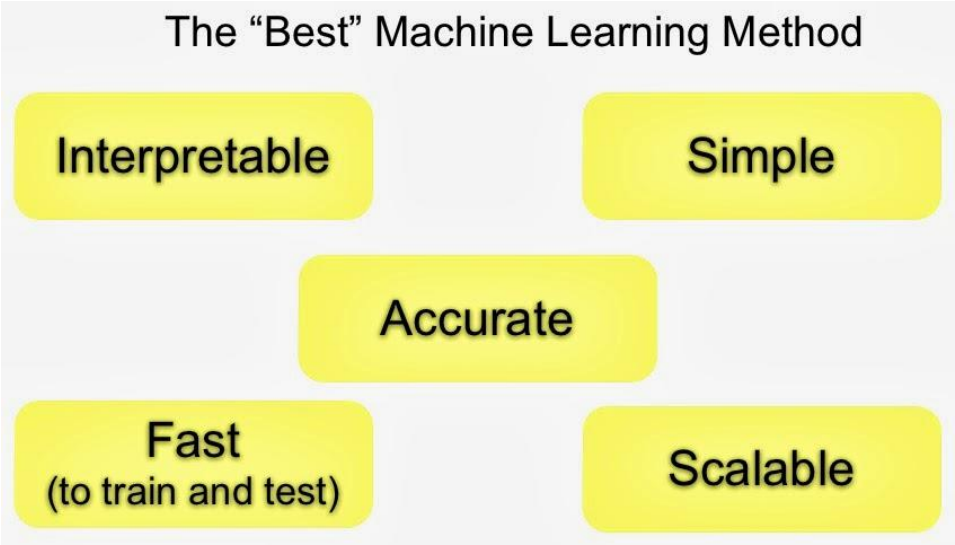
Classification Measures – Imbalanced Training Set Some solutions

- 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

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

Permalink.

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.*

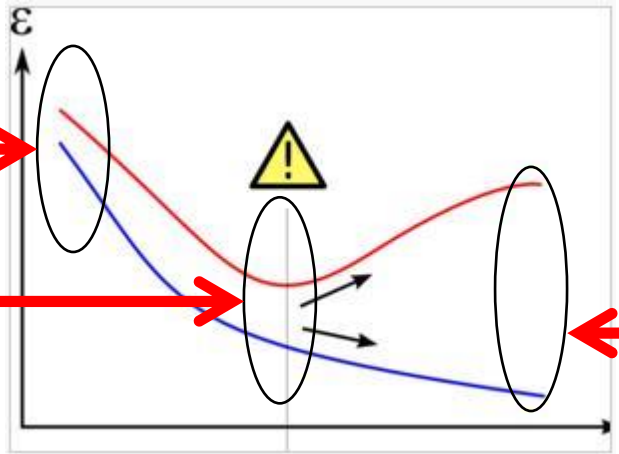
Neat. But the winning prize? Eh... just not worth it:

> *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/

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.*



Underfitting

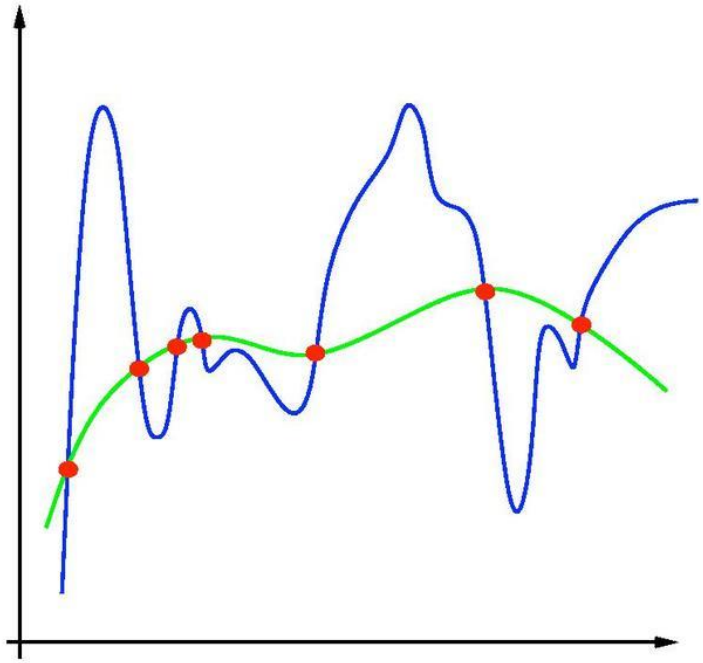
Just right

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

Overfitting

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

Overfitting



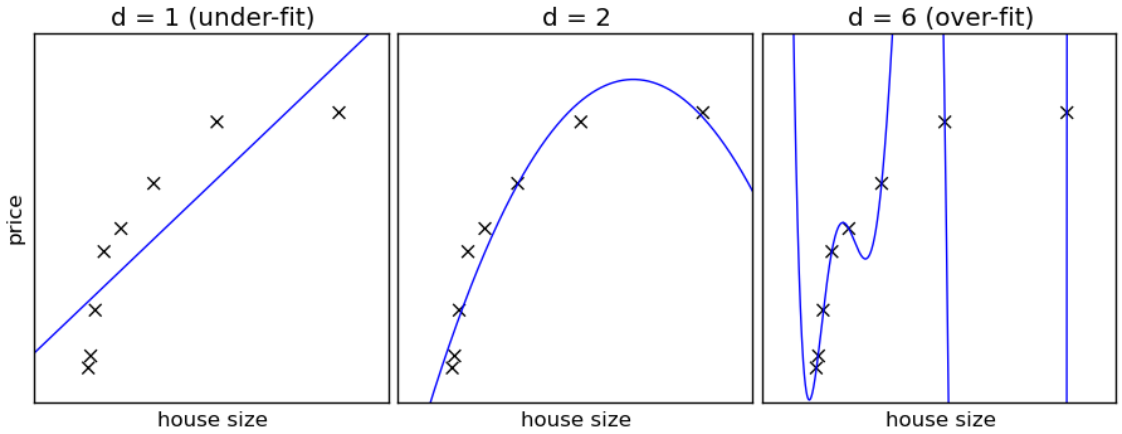
(by Tomaso Poggio, http://www.mit.edu/~9.520/spring12/slides/class02/class02.pdf)

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

- 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**

δ(f(x),h(x))=1 if f(x)≠h(x)

δ(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.

Sample Error as Estimator

- Q1: What is the best estimate of the accuracy over future examples drawn from the same distribution?
- The best estimate of the true error is the sample error.
- Proof can be found in Mitchell's book (chapter 5). Not examinable.

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.

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)

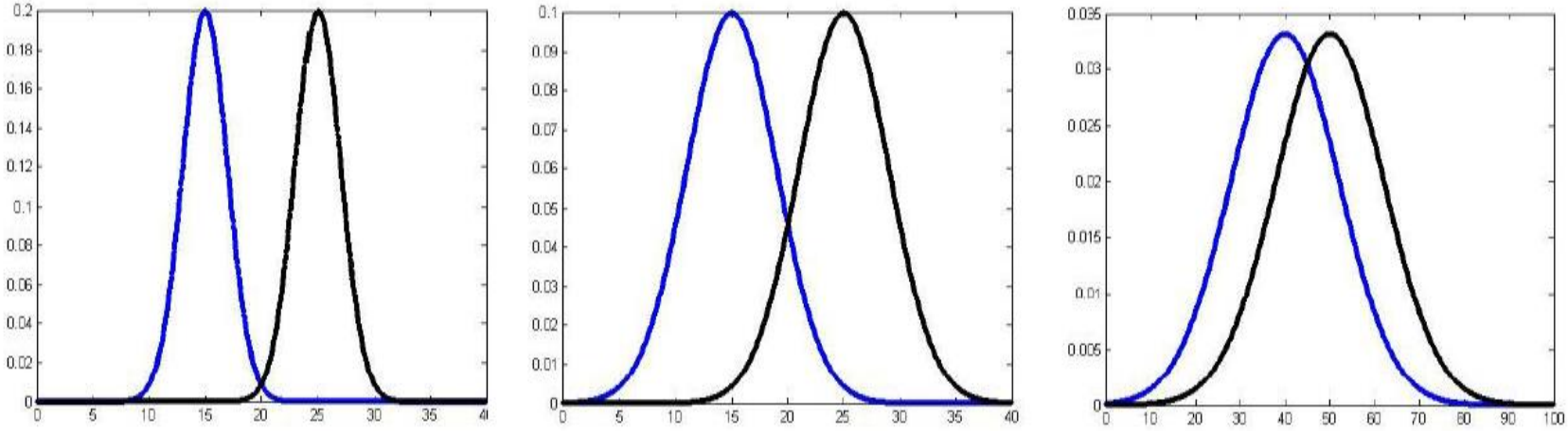
Given that $error_s(h)$=0.22 and n= 50, and z_N=1.96 for N = 95%, we can now say that with 95% confidence $error_D(h)$ will lie in the interval:

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

$$[0.11, 0.34]$$

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

Comparing Two Algorithms



- Consider the distributions of 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.

Statistical Tests

- There are several statistical tests: T-test, Wilcoxon, Randomisation etc.
- A set of observations **x** and **y** (e.g. classification error) for each algorithm are needed.
- Two-sample T-test: **x, y** could be the classification errors on two different datasets
- Paired T-test: **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.
- The t-test tells us if the means of the two sets are significantly different.