Transcripts – Machine Learning in Practice (2)

1. (it was lect 9 in 2015).
2. Acknowledgements
3. Outline: Comparing schemes: the t-test, Predicting probabilities, Cost-sensitive measures, Occam's razor
4. We often need to compare two different learning methods on the same problem to see which is the better one to use. It seems simple: estimate the error using cross-validation (or any other suitable estimation procedure), perhaps repeated several times, and choose the scheme whose estimate is smaller. This is quite sufficient in many practical applications: if one method has a lower estimated error than another on a particular dataset, the best we can do is to use the former method's model. However, it may be that the difference is simply caused by estimation error, and in some circumstances it is important to determine whether one scheme is really better than another on a particular problem. This is a standard challenge for machine learning researchers, who have to show that any improvement is not just a chance effect in the estimation process.
5. If there were unlimited data, we could use a large amount for training and evaluate performance on a large independent test set, obtaining confidence bounds just as before. However, if the difference turns out to be significant we must ensure that this is not just because of the particular dataset we happened to base the experiment on. What we want to determine is whether one scheme is better or worse than another on average, across all possible training and test datasets that can be drawn from the domain. Because the amount of training data naturally affects performance, all datasets should be the same size: indeed, the experiment might be repeated with different sizes to obtain a learning curve. For the moment, assume that the supply of data is unlimited. For definiteness, suppose that cross-validation is being used to obtain the error estimates (other estimators, such as repeated cross-validation, are equally viable). For each learning method we can draw several datasets of the same size, obtain an accuracy estimate for each dataset using cross-validation, and compute the mean of the estimates. Each cross-validation experiment yields a different, independent error estimate. What we are interested in is the mean accuracy across all possible datasets of the same size, and whether this mean is greater for one scheme or the other. From this point of view, we are trying to determine whether the mean of a set of samples is significantly greater than, or significantly less than, the mean of another. This is a job for a statistical device known as the t-test, or Student's t-test.
6. break
7. When we have limited data and a limited number of estimates for computing the mean we use the t-test. The t-test is also called Student's t-test. The *t-test* tells us whether the means of two samples are significantly different. In this video clip we are going to describe in detail a version of the t-test called paired t-test. This version is called "paired" because we compare matched pairs of results for each dataset. Basically this means that we use the same datasets to compare 2 different learning schemes.
8. Let's see how it works. We take a first set of samples -- from x-sub-1 to x-sub-k -- obtained by successive 10-fold-cross-validation using one learning scheme. Then, we take a second set of samples, -- from y-sub-1 to y-sub-k -- obtained by successive 10-fold cross-validations using another learning scheme. Each cross-validation estimate

is generated using a different dataset, but all the datasets are of the same size and from the same domain. m-sub-x represents the mean of the first set of samples and m-sub-y represents the mean of the second set of sample. Now, we know that if had enough samples, the mean of a set of independent samples is normally distributed. If we knew the variance of that normal distribution – so that it could be reduced to have 0 mean and unit variance – we could obtain confidence limits, as we described in the previous lecture. However the variance is unknown, but we can obtain it by estimating it from the set of samples. The variance of mean m-sub-x can be estimated by dividing by k the variance calculated from the samples x-sub-1 up to x-sub-k. k is the total number of samples. We call the variance calculated from the samples x-sub-1 up to x-sub-k: small sigma sub x squared. We compute the variance also for the set of samples y. Now we use the greek letter mi to represent the true mean of a set of independent samples with normal distribution. If mi-sub-x and mi-sub-y are the true means then we reduce the distribution of m-sub-x to have 0 mean and a unit variance by using the formulas on the RHS.

9. Because the variance is only an estimate, it does not necessarily have a normal distribution (although the distribution is normal for very large samples). Instead, it has what is called a Student's distribution with k - 1 degrees of freedom. What this means in practice is that we have to use a table of confidence intervals for Student's distribution rather than the confidence table for the normal distribution that we saw in the previous lecture. For 9 degrees of freedom (which is the correct number here if we are using the average of 10 cross-validations) the appropriate confidence limits are shown on the screen in the table on the LHS. If you compare these values with those in the table on the RHS (that shows the degrees of freedom for normal distribution), you will see that the Student's figures are slightly more conservative, and this reflects the additional uncertainty caused by having to estimate the variance. Different tables are needed for different numbers of degrees of freedom, and if there are more than 100 degrees of freedom the confidence limits are very close to those for the normal distribution. The figures in the tables on the screen are for a "one-sided" confidence interval.
10. To decide whether the means m-sub-x and m-sub-y, each an average of the same number k of samples, are the same or not, we consider the differences m-sub-d between corresponding observations, so m-sub-d = m-sub-x minus m-sub-y. This is legitimate because the observations are paired. The mean of this difference is just the difference between the two means m-sub-x minus m-sub-y, and, like the means themselves, it has a Student's distribution with k - 1 degrees of freedom. If the means are the same, the difference is zero (this is called the **null hypothesis**); if they're significantly different, the difference will be significantly different from zero. So for a given confidence level, we will check whether the actual difference exceeds the confidence limit. First, we reduce the difference to a zero-mean, unit-variance variable called the t-statistic: you can see the formula on the screen.
11. Then, we decide on a confidence level—generally, 5% or 1% is used in practice. From this the confidence limit z is determined using the table that we saw before on the LHS if k is 10; if it is not, a confidence table of the Student's distribution for the k value in question is used. A two-tailed test is appropriate because we do not know in advance whether the mean of the x's is likely to be greater than that of the y's or vice versa: thus for a 1% test we use the value corresponding to 0.5% (in the table). If the value of t according to the preceding formula is greater than z, or less than -z, we **reject the null hypothesis that the means are the same and conclude that there**

really is a significant difference between the two learning methods on that domain for that dataset size.

12. What happens if the observations were not paired? What happens if the number of datasets for each scheme is not the same? Then it is necessary to use a regular, non-paired t-test. If the means are normally distributed, as we are assuming, the difference between the means is also normally distributed. Instead of taking the mean of the difference, m-sub-d, we take the difference of the means, m-sub-x minus m-sub-y . Of course, that's the same thing: the mean of the difference is the difference of the means. But the variance of the difference m-sub-d is not the same. If the variance of the samples x-sub-1 to x-sub-k is **small sigma-sub-x squared** and the variance of the samples y-sub-1 to y-sub-j is **small sigma-sub-y squared**, the best estimate of the variance of the difference of the means is calculated using the formula you see on the screen. It is this variance (or rather, its square root) that should be used as the denominator of the t-statistic given previously.
13. break
14. So far, we have assumed that the goal is to maximize the success rate of the predictions. The outcome for each test instance is either correct, if the prediction agrees with the actual value for that instance, or incorrect, if it does not. There are no greys: everything is black or white, correct or incorrect. In many situations, this is the most appropriate perspective. If the learning scheme, when it is actually applied, results in either a correct or an incorrect prediction, success is the right measure to use. This is called a 0 - 1 loss function: the "loss" is either zero if the prediction is correct or one if it is not. We have already come across 0-1 loss when we presented decision trees. Other situations are less black and white. Most learning methods can associate a probability with each prediction (as the Naïve Bayes method does). It might be more natural to take this probability into account when judging correctness. For example, a correct outcome predicted with a probability of 99% should perhaps weigh more heavily than one predicted with a probability of 51%. Whether it is appropriate to take prediction probabilities into account depends on the application.
15. Suppose that for a single instance there are k possible outcomes, or classes, and for a given instance the learning scheme comes up with a probability vector from p-sub-1 to p-sub-k for the classes (where these probabilities sum to 1). The actual outcome for that instance will be one of the possible classes. However, it is convenient to express it as a vector from a-sub-1 to a-sub-k whose c-th component is 1 and all other components are 0. The c-th component is the actual class. We can express the penalty associated with this situation as a loss function that depends on both the p vector and the on the a vector. One criterion that is frequently used to evaluate probabilistic prediction is the quadratic loss function: the sigma sub j of p-sub-j minus a-sub-j squared refers to a single instance. Note that the summation is over possible outputs not over different instances. Just one of the a's will be 1 and the rest will be 0. When the test set contains several instances, the loss function is summed over them all. Minimizing the squared error is a difficult problem. In the present context, the quadratic loss function gives preference to predictors that are able to make the best guess at the true probabilities. I skip all the details here and I just point out that the quadratic loss function is frequently used to measure the success rate probabilistic prediction situations.
16. Another popular criterion for the evaluation of probabilistic prediction is the informational loss function: is –log(p-sub-c), where the cth prediction is the correct

one. We are already familiar with the negative logarithm since we talked about entropy in lecture 4. You can see the formula on the screen. Because probabilities are always less than 1, their logarithms are negative, and the minus sign makes the outcome positive.

17. Which of the loss functions should you use? That's a good question, and there is no universally agreed-upon answer. The quadratic loss function takes account not only of the probability assigned to the event that actually occurred, but also the other probabilities. For example, in a four-class situation, suppose you assigned 40% to the class that actually came up and distributed the remainder among the other three classes. The quadratic loss will depend on how you distributed. The loss will be smallest if the 60% was distributed evenly among the three classes: an uneven distribution will increase the sum of the squares. The informational loss function, on the other hand, depends solely on the probability assigned to the class that actually occurred. If you're gambling on a particular event coming up, and it does, who cares how you distributed the remainder of your money among the other events? If you assign a very small probability to the class that actually occurs, the information loss function will penalize you massively.
18. Break
19. Now let's see how chance can affect the success rate. Let's take a multiclass example. In multiclass prediction, the result on a test set is displayed as a two-dimensional confusion matrix with a row and column for each class. Each matrix element shows the number of test examples for which the actual class is the row and the predicted class is the column. Good results correspond to large numbers down the main diagonal, ideally off-diagonal elements should be 0 or should contain small numbers. The table on the LHS shows the predictions of a classifier on 3 classes. Our test set has 200 instances and 140 of them are predicted correctly, so the success rate is 70%. But is this a fair measure of overall success? How many agreements occur by chance? Our classifier predicts a total of 120 a's, 60 b's, and 20 c's; On the RHS you can see the results achieved by a classifier that guesses randomly. The random classifier predicts exactly the same number of a's, b's and c's, and this random classifier gets 60 + 18 + 4 = 82 instances correct . A measure called the Kappa statistic takes this random guessing into account by deducting it from the actual classifier's success rate. Let's see how.
20. The actual numbers of the test set are 100 "a's", 60 "b's" and 40 "c's": Proportions of the class "a" = 0.5 (ie 100 instances out of 200 à 50% à 50/100 à 0.5) Proportions of the class "b" = 0.3 (ie 60 instances out of 200 à 30% à 30/100 à 0.3) Proportions of the class "c" = 0.2 (ie 40 instances out of 200 à 20% à 20/100 à 0.2) Both classifiers (see below) returns 120 a's, 60 b's and 20 c's, but one classifier is random. How much the actual classifier improves on the random classifier? A classifier randomly guessing would return the predictions in the table on the RHS: 0.5*120=60; 0.3*60=18; 0.2*20=4 à 60+18+4 = 82 The actual classifier returns the predictions in the table on the LHS, 140 correct predictions (see diagonal), ie 70% success rate: When we compute the k statistic we subtract the random guessing from the actual guessing by applying the formula you see on the screen, that is observed predictions minus random predictions divided by the total number of instances in the test set minus the random predictions. So we get = 140-82/200-82 = 58/118=0.49=49%; So the actual success rate of 70% represents an improvement of 49% on random guessing!

21. A k statistic of 100% (or 1) implies a perfect classifier. A k statistic of 0 implies that the classifier provides no information and behaves as if it were guessing randomly. In summary, the Kappa statistic is used to measure the agreement between predicted and observed categorizations of a dataset, and corrects the agreement that occurs by chance. Weka provides the k statistic value to assess the success rate beyond the chance. However, like the plain success rate, it does not take costs into account.
22. Break
23. The evaluations that have been discussed so far do not take into account the cost of making wrong decisions, wrong classifications. Evaluation by classification accuracy and k statistic assumes that all types of errors have the same cost. But this is not the case in real-world scenarios. For instance, what is the cost of sending promotional mail to a household that doesn't respond is far less than the lost-business cost of not sending it to a household that would have responded. So the cost of these two types of errors are different: one cost is lower than the other. The same is true if you have to detect a potential terrorist or if a bank has to approve a loan, and so on.
24. In the two-class case with classes yes and no, such as approve a loan to a customer, a single prediction has the four different possible outcomes shown in the table on the screen. The true positives (TP) and true negatives (TN) are correct classifications. A false positive (FP) occurs when the outcome is incorrectly predicted as yes (or positive) when it is actually no (negative). A false negative (FN) occurs when the outcome is incorrectly predicted as negative when it is actually positive. The true positive rate is TP divided by the total number of positives, which is TP + FN; the false positive rate is FP divided by the total number of negatives, FP + TN. The overall success rate is the number of correct classifications divided by the total number of classifications: Finally, the error rate is one minus this.
25. If the costs are known, they can be incorporated into the decision-making process. In the two-class case, in which the confusion matrix is like that of the table on the screen, the two kinds of error—false positives and false negatives— will have different costs; likewise, the two types of correct classification may have different benefits. In the two-class case, costs can be summarized in the form of a 2 * 2 matrix in which the diagonal elements represent the two types of correct classification and the off-diagonal elements represent the two types of error. In the multiclass case this generalizes to a square matrix whose size is the number of classes, and again the diagonal elements represent the cost of correct classification. Taking the cost matrix into account replaces the success rate by the average cost.
26. 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.
27. Break
28. In practice, costs are rarely known with any degree of accuracy, and people will want to analyse various scenarios. Imagine you're in the direct mailing business and are contemplating a mass mailout of a promotional offer to 1,000,000 households—most of whom won't respond, of course. Let us say that, based on previous experience, the proportion who normally respond is known to be 0.1% (1000 respondents). Suppose a data mining tool is available that, based on known information about the

households, identifies a subset of 100,000 for which the response rate is 0.4% (400 respondents). It may well pay off to restrict the mailout to these 100,000 households—that depends on the mailing cost compared with the return gained for each response to the offer.

29. Given a learning method that outputs probabilities for the predicted class of each member of the set of test instances (as Naïve Bayes does), your job is to find subsets of test instances that have a high proportion of positive instances, higher than in the test set as a whole. To do this, the instances should be sorted in descending order of predicted probability of yes. Then, to find a sample of a given size with the greatest possible proportion of positive instances, just read the requisite number of instances off the list, starting at the top. If each test instance's class is known, you can calculate the lift factor by simply counting the number of positive instances that the sample includes, dividing by the sample size to obtain a success proportion and dividing by the success proportion for the complete test set to determine the lift factor. The table on the screen shows an example for a small dataset with 150 instances, of which 50 are yes responses—an overall success proportion of 33%. The instances have been sorted in descending probability order according to the predicted probability of a yes response. The first instance is the one that the learning scheme thinks is most likely to be positive, the second is the next most likely, and so on. The numeric values of the probabilities are unimportant: rank is the only thing that matters. With each rank is given the actual class of the instance.
30. Thus the learning method was right about items 1 and 2—they are indeed positives—but wrong about item 3, which turned out to be a negative. Now, if you were seeking the most promising sample of size 10 but only knew the predicted probabilities and not the actual classes, your best bet would be the top ten ranking instances. Eight of these are positive, so the success proportion for this sample is 80%, corresponding to a lift factor of four.
31. If you knew the different costs involved, you could work them out for each sample size and choose the most profitable. But a graphical depiction of the various possibilities will often be far more revealing than presenting a single "optimal" decision. Repeating the preceding operation for different-sized samples allows you to plot a lift chart like that of Figure 5.1. The horizontal axis shows the sample size as a proportion of the total possible mailout. The vertical axis shows the number of responses obtained. The lower left and upper right points correspond to no mailout at all, with a response of 0, and a full mailout, with a response of 1000. The diagonal line gives the expected result for different- sized random samples. But we do not choose random samples; we choose those instances which, according to the data mining tool, are most likely to generate a positive response. These correspond to the upper line, which is derived by summing the actual responses over the corresponding percentage of the instance list sorted in probability order. The two particular scenarios described previously are marked: a 10% mailout that yields 400 respondents and a 40% one that yields 800. Where you'd like to be in a lift chart is near the upper left-hand corner: at the very best, 1000 responses from a mailout of just 1000, where you send only to those households that will respond and are rewarded with a 100% success rate. Any selection procedure worthy of the name will keep you above the diagonal— otherwise, you'd be seeing a response that was worse than for random sampling. So the operating part of the diagram is the upper triangle.
32. Break

33. ROC curves are similar to lift charts. ROC stands for "receiver operating characteristic" .ROC curves are used in signal detection to show tradeoff between hit rate and false alarm rate over noisy channel. ROC curves depict the performance of a classifier without regard to class distribution or error costs. They plot the number of positives included in the sample on the vertical axis, expressed as a percentage of the total number of positives, against the number of negatives included in the sample, expressed as a percentage of the total number of negatives, on the horizontal axis. The vertical axis is the same as that of the lift chart except that it is expressed as a percentage. The horizontal axis is slightly different— number of negatives rather than sample size.
34. The figure on the screen shows an example ROC curve for the sample of test data in the table. The roc curve is the jagged line. You can follow the curve along with the table. From the origin, [look at the column: Actual class] go up two (two positives), along one (one negative), up five (five positives), along one (one negative), up one, along one, up two, and so on. Each point corresponds to drawing a line at a certain position on the ranked list, counting the yes's and no's above it, and plotting them vertically and horizontally, respectively. As you go farther down the list, corresponding to a larger sample, the number of positives and negatives both increase.
35. The jagged ROC line in the previous slide depends on the details of the particular sample of test data. This sample dependence can be reduced by applying cross-validation. For each different number of no's—that is, each position along the horizontal axis—take just enough of the highest-ranked instances to include that number of no's, and count the number of yes's they contain. Finally, average that number over different folds of the cross-validation. **The result is a smooth curve like that in the previous slide.** This is just one way of using cross-validation to generate ROC curves. This method is implemented in WEKA. However, this is just one possibility. Another possibility is to generate an ROC curve for each fold and average them.
36. It is useful to look at ROC curves obtained using different learning schemes. Look at the screen: the method A excels if a small, focused sample is sought; that is, if you are working toward the left-hand side of the graph. Clearly, if you aim to cover just 40% of the true positives you should choose method A, which gives a false positive rate of around 5%, rather than method B, which gives more than 20% false positives. But method B excels if you are planning a large sample: if you are covering 80% of the true positives, method B will give a false positive rate of 60% as compared with method A's 80%. The shaded area is called the convex hull of the two curves, and you should always operate at a point that lies on the upper boundary of the convex hull. Only if a particular scheme generates a point that lies on the convex hull should it be used alone: otherwise, it would always be better to use a combination of classifiers corresponding to a point that lies on the convex hull. To summarize ROC curves in a single quantity, people sometimes use the area under the curve (AUC) because, roughly speaking the larger the area the better the model. The area also has a nice interpretation as the probability that the classifier ranks a randomly chosen positive instance above a randomly chosen negative one.
37. break
38. Given a query, a Web search engine produces a list of hits that represent documents supposedly relevant to the query. Compare one system that locates 100 documents, 40 of which are relevant, with another that locates 400 documents, 80 of which are

relevant. Which is better? The answer should now be obvious: it depends on the relative cost of false positives, documents that are returned that aren't relevant, and false negatives, documents that are relevant that aren't returned. Information retrieval researchers define parameters called recall and precision. Information retrieval experts use recall–precision curves that plot one against the other, for different numbers of retrieved documents, in just the same way as ROC curves and lift charts. For example, if the list of yes's and no's in the previous table represented a ranked list of retrieved documents and whether they were relevant or not, and the entire collection contained a total of 40 relevant documents, then "recall at 10" would refer to recall for the top ten documents, that is, 8/40 = 5%; while "precision at 10" would be 8/10 = 80%. Although such measures may be useful if costs and class distributions are unknown and one method must be chosen to handle all situations, no single number is able to capture the tradeoff. That can only be done by two-dimensional depictions such as lift charts, ROC curves, and recall–precision diagrams.

39. Occam's (or Ockham's) razor is a principle attributed to the 14th century logician and Franciscan friar William of Ockham. Ockham was the village in the English county of Surrey where he was born. The principle states that "Entities should not be multiplied unnecessarily." Many scientists have adopted or reinvented Occam's Razor, The most useful statement of the principle for scientists is "when you have two competing theories that make exactly the same predictions, the simpler one is the better." Occam's razor shaves philosophical hairs off a theory. The idea is that the best scientific theory is the smallest one that explains all the facts. If what is learned is a theory, then the errors it makes are like exceptions to the theory.
40. Say that you have to choose between 2 models: Model 1: very simple and elegant that accounts for the data almost perfectly. Model 2: significantly more complex model that reproduces the data without mistakes. According to Occam's principle, Model 1 is preferable.
41. The end

Statistical Learning: Interval Estimation.

Hypothesis Testing & Statistical Significance (daume' iii: 63-64)

1) Suppose we measure the error of a classifier on a test set and obtain a certain numerical error rate, say 25%. This corresponds to a success rate of 75%. This is only an estimate on a sample (our dataset). What can we say about the "true" success rate on the target population? All we can say that it is supposed to be close to 75%. But how close? within 5% or 10% or...?.

2) To answer these questions, we need some statistical reasoning. In statistics, a succession of independent events that either succeed or fail is called a Bernoulli process. The classic example is coin tossing. Each toss is an independent event. Let's say we always predict heads; but rather than "heads" or "tails," each toss is considered a "success" or a "failure." Let's say the coin is biased, but we don't know what the probability of heads is. Then, if we actually toss the coin 100 times and 75 of them are heads, we have a situation much like the one described previously for a classifier with an **observed/estimated** 75% success rate on a test set. What can we say about the **true** success probability?

3) In other words, imagine that there is a Bernoulli process—a biased coin—whose true (but unknown) success rate is **p**. So p is true value that we are looking for. Suppose that out of N trials, S are successes: thus the observed success rate is **f** = S/N. f is the success rate that we observe. The question is, what does this f tell us about the true success rate **p**? The answer to this question is usually expressed as a confidence interval; that is, **p** *lies within a certain specified interval with a certain specified confidence*. For example, if S = 750 successes are observed out of N = 1000 trials, this indicates that the true success rate must be around 75%. But how close to 75%? It turns out that with 80% confidence, the true success rate p lies between 73.2% and 76.7%. If S = 75 successes are observed out of N = 100 trials, this also indicates that the true success rate must be around 75%. But the experiment is smaller, and the 80% confidence interval for p is wider, stretching from 69.1% to 80.1%. Small samples produce larger intervals and which means more uncertainty.
4) Weka book: section 5.2 explains the calculations. But I simplify and say that confidence intervals are calculated based on the standard error of a measurement. Generally, the larger the number of measurements made, basically the larger the sample, the smaller the standard error and narrower the resulting confidence intervals. In other words, standard error is a statistical term that measures the accuracy with which a sample represents a population. In statistics, a **sample mean** deviates from the **true mean** of a population; this deviation is the standard error. You can see the formula on the screen. SEM = standard error of the mean; s = sample standard deviation ; n = size (number of observations) of the sample. You can also see the the sample standard deviation formula (we have already seen in lect 2) where: s = sample standard deviation; x1, ..., xN = the sample data set ; $\bar{x}$ = mean value of the sample data set ; N = size of the sample data set. The standard error formula shows that the larger the sample size, the smaller the standard error of the mean. More specifically, the size of the standard error of the mean is inversely proportional to the square root of the sample size.

5) Once the standard error is calculated, the confidence interval is determined by multiplying the standard error by a constant that reflects the level of significance desired, based on the normal distribution. The constant for 80 percent confidence intervals is 1.28.
6) This result can be used to obtain the values in the preceding numeric example. Setting f = 75%, N = 1000, and c = 80% (so that z = 1.28) leads to the interval [0.732,0.767] for p, and N = 100 leads to [0.691,0.801] for the same level of confidence. Note that the normal distribution assumption is only valid for large N (say, N > 100). Thus f = 75% and N = 10 leads to confidence limits [0.549,0.881]—but these should be taken with a grain of salt. $\in$ set membership=is an element of.

----- ------ ---- (slides from former Lect 8 2015: evaluation, bootstrap

- There are lots of models that we can use to learn how to predict the class of new, previously unseen instances. We practiced with a couple of them during the lab sessions. But to determine which ones to use on a particular problem we need systematic ways to **evaluate**, ie to assess, how different methods work and to compare one with another. Evaluation is a fundamental part of machine learning.

Evaluation can be tricky. What's the problem? We have the training set; surely we can just look at how well different methods do on that. We have already emphasised in previous lectures that we need ways of predicting the performance of learning method based on experiments with data independent from the training set. When a vast supply of data is available, this is no problem: just make a model based on a large training set, and try it out on another large test set. But although there is a lot of buzz about "big data"— it is often the case that data, quality data, annotated data, the labelled data that we need for the evaluation is scarce. The question of predicting performance based on limited data is important and still controversial. We will encounter many different techniques, of which one—**repeated cross-validation**—is probably the evaluation method chosen in most practical limited-data situations.

- Comparing the performance of different machine learning methods on a given problem is another matter that is not easy either. It arises when we want to compare the performance of 2 or more learning methods on the same type of data. In practice this means that we must make sure that apparent differences are not caused by chance effects, and we need to use statistical tests. Additionally, in some situations we need to predict the class probabilities rather than the classes themselves. Then we need to notice that not all the error have the same importance. Each type of misclassification has its own cost. The cost of an error can be different if, for example, a positive example was erroneously classified as negative or vice versa. Each of these different cases require different techniques that we are going to describe in this part of the lecture.
- For classification problems, it is natural to measure a classifier's performance in terms of the error rate. The classifier predicts the class of each instance: if it is correct, that is counted as a success; if not, it is an error. The error rate is just the proportion of errors made over a whole set of instances, and it measures the overall performance of the classifier. Of course, what we are interested in is the future performance on new data, not the past performance on old data. We already know the classifications of each instance in the training set, which after all is why we can use it for training. So the question is, is the error rate on old data likely to be a good indicator of the error rate on new data? We know already that the answer is : no— not if the old data was used during the learning process to train the classifier. Why? Because the classifier has been learned from the very same training data, any estimate of performance based on that data will be optimistic. The error rate on the training data is called the **empirical error**, or training error or resubstitution error. Different books uses different terms, but we stick to empirical error. Although it is not a reliable predictor of the true error rate on new data, the empirical error is nevertheless often useful to know, for ex when you want to have an idea of the consistency of the classes.
- To predict the performance of a classifier on new data, we need to assess its error rate on a dataset that played no part in the formation of the classifier. This independent dataset is called the test set. We assume that both the training data and the test data are representative samples of the underlying problem. It is important that the test data was not used in any way to create the classifier. For example, some learning methods involve two stages, one to come up with a basic learning model, and the second stage to optimize parameters. But none of the 2 sets of data used in these 2 stages can be used to determine an estimate of the future error rate. In such situations people often talk about three datasets: the training data, the validation data (aka development set), and the test data.

- The training data is used by one or more learning methods to come up with classifiers. The validation data is used to optimize parameters of those classifiers, or to select a particular one. Then the test data is used to calculate the error rate of the final, optimized, method. Each of the three sets must be chosen independently: the validation set must be different from the training set to obtain good performance in the optimization or selection stage, and the test set must be different from both to obtain a reliable estimate of the true error rate. The fundamental assumption is that all the sets are representative samples of the same population and distribution. But of course the sets may differ and have different sources. For example, the set can be about customer data coming from different towns or states.
- It may be that once the error rate has been determined, the test data is bundled back into the training data to produce a new classifier for actual use. There is nothing wrong with this: it is just a way of maximizing the amount of data used to generate the classifier that will actually be employed in practice. What is important is that error rates are not based on any of this data. Generally, the larger the training sample the better the classifier, although the returns begin to diminish once a certain volume of training data is exceeded. And the larger the test sample, the more accurate the error estimate. The accuracy of the error estimate can be quantified statistically, as we will see it shortly. The real problem occurs when there is not a vast supply of data available. In many situations the training data must be classified manually—and so must the test data, of course, to obtain error estimates. This limits the amount of data that can be used for training, validation, and testing, and the problem becomes how to make the most of a limited dataset. From this dataset, a certain amount is held over for testing—this is called the holdout procedure—and the remainder is used for training (and, if necessary, part of that is set aside for validation). There's a dilemma here: to find a good classifier, we want to use as much of the data as possible for training; to obtain a good error estimate, we want to use as much of it as possible for testing. We will shortly see methods that help us deal with this dilemma.
- Break….
- break
- Now consider what to do when the amount of data for training and testing is limited. The holdout method reserves a certain amount for testing and uses the remainder for training (and sets part of that aside for validation, if required). In practical terms, it is common to hold out one-third of the data for testing and use the remaining two-thirds for training. Of course, you may be unlucky: the sample used for training (or testing) might not be representative. In general, you cannot tell whether a sample is representative or not. But there is one simple check that might be worthwhile: each class in the full dataset should be represented in about the right proportion in the training and testing sets. If, by bad luck, all examples with a certain class were missing from the training set, you could hardly expect a classifier learned from that data to perform well on the examples of that class—and the situation would be exacerbated by the fact that the class would necessarily be overrepresented in the test set because none of its instances made it into the training set! Instead, you should ensure that the random sampling is done in such a way as to guarantee that each class is properly represented in both training and test sets. This procedure is called stratification, and we might speak of stratified holdout. Although it is generally well worth doing, stratification provides only a primitive safeguard against uneven representation in training and test sets.

- A more general way to mitigate any bias caused by the particular sample chosen for holdout is to repeat the whole process, training and testing, several times with different random samples. In each iteration a certain proportion— say two-thirds—of the data is randomly selected for training, possibly with stratification, and the remainder used for testing. The error rates on the different iterations are averaged to yield an overall error rate. This is the repeated holdout method of error rate estimation. In a single holdout procedure, you might consider swapping the roles of the testing and training data—that is, train the system on the test data and test it on the training data—and average the two results, thus reducing the effect of uneven representation in training and test sets.
- Unfortunately, this is only really plausible with a 50 : 50 split between training and test data, which is generally not ideal—it is better to use more than half the data for training even at the expense of test data. However, a simple variant forms the basis of an important statistical technique called cross-validation. In cross-validation, you decide on a fixed number of folds, or partitions of the data. Suppose we use three. Then the data is split into three approximately equal partitions and each in turn is used for testing and the remainder is used for training. That is, use two-thirds for training and one-third for testing and repeat the procedure three times so that, in the end, every instance has been used exactly once for testing. This is called threefold cross-validation, and if stratification is adopted as well—which it often is—it is stratified threefold cross-validation.
- The standard way of predicting the error rate of a learning technique given a single, fixed sample of data is to use stratified 10-fold cross-validation. The data is divided randomly into 10 parts in which the class is represented in approximately the same proportions as in the full dataset. Each part is held out in turn and the learning scheme trained on the remaining nine-tenths; then its error rate is calculated on the holdout set. Thus the learning procedure is executed a total of 10 times on different training sets (each of which have a lot in common). Finally, the 10 error estimates are averaged to yield an overall error estimate. Why 10? Extensive tests on numerous datasets, with different learning techniques, have shown that 10 is about the right number of folds to get the best estimate of error, and there is also some theoretical evidence that backs this up. A single 10-fold cross-validation might not be enough to get a reliable error estimate. Different 10-fold cross-validation experiments with the same learning method and dataset often produce different results, because of the effect of random variation in choosing the folds themselves. Stratification reduces the variation, but it certainly does not eliminate it entirely. When seeking an accurate error estimate, it is standard procedure to repeat the cross-validation process 10 times—that is, 10 times 10-fold cross-validation—and average the results. This involves invoking the learning algorithm 100 times on datasets that are all nine-tenths the size of the original.
- Obtaining a good measure of performance is a computation-intensive undertaking. Leave-one-out cross-validation is simply n-fold cross-validation, where n is the number of instances in the dataset. Each instance in turn is left out, and the learning method is trained on all the remaining instances. It is judged by its correctness on the remaining instance—one or zero for success or failure, respectively. The results of all n judgments, one for each member of the dataset, are averaged, and that average represents the final error estimate. This procedure is an attractive one for two reasons. First, the greatest possible amount of data is used for training in each case, which presumably increases the chance that the classifier is an accurate one.

Second, the procedure is deterministic: no random sampling is involved. There is no point in repeating it 10 times, or repeating it at all: the same result will be obtained each time. Set against this is the high computational cost, because the entire learning procedure must be executed n times and this is usually quite infeasible for large datasets. Nevertheless, leave-one-out seems to offer a chance of squeezing the maximum out of a small dataset and obtaining as accurate an estimate as possible.

- But there is a disadvantage to leave-one-out cross-validation, apart from the computational expense. By its very nature, it cannot be stratified—worse than that, it guarantees a non stratified sample. Stratification involves getting the correct proportion of examples in each class into the test set, and this is impossible when the test set contains only a single example. A dramatic, although highly artificial, illustration of the problems this might cause is to imagine a completely random dataset that contains the same number of each of two classes. The best that an inducer can do with random data is to predict the majority class, giving a true error rate of 50%. But in each fold of leave-one-out, the opposite class to the test instance is in the majority—and therefore the predictions will always be incorrect, leading to an estimated error rate of 100%!
- break
- The bootstrap is based on the statistical procedure of sampling with replacement. Previously, whenever a sample was taken from the dataset to form a training or test set, it was drawn without replacement. That is, the same instance, once selected, could not be selected again. The idea of the bootstrap is to sample the dataset with replacement to form a training set. We will describe a particular variant called the 0.632 bootstrap. For this, a dataset of n instances is sampled n times, with replacement, to give another dataset of n instances. Because some elements in this second dataset will (almost certainly) be repeated, there must be some instances in the original dataset that have not been picked: we will use these as test instances.
- What is the chance that a particular instance will not be picked for the training set? It has a 1/n probability of being picked each time and therefore a 1 - 1/n probability of not being picked. Multiply these probabilities together according to the number of picking opportunities, which is n, and the result is : 0.368. This gives the chance of a particular instance not being picked at all. Thus for a reasonably large dataset, the test set will contain about 36.8% of the instances and the training set will contain about 63.2% of them (now you can see why it's called the 0.632 bootstrap). Some instances will be repeated in the training set, bringing it up to a total size of n, the same as in the original dataset.
- The figure obtained by training a learning system on the training set and calculating its error over the test set will be a pessimistic estimate of the true error rate, because the training set, although its size is n, nevertheless contains only 63% of the instances, which is not a great deal compared, for example, with the 90% used in 10-fold cross-validation. To compensate for this, we combine the test-set error rate with the resubstitution error on the instances in the training set. The resubstitution figure gives a very optimistic estimate of the true error and should certainly not be used as an error figure on its own. But the bootstrap procedure combines it with the test error rate to give a final estimate e as you can see on the screen. Then, the whole bootstrap procedure is repeated several times, with different replacement samples for the training set, and the results averaged.

- The bootstrap procedure may be the best way of estimating error for very small datasets. However, like leave-one-out cross-validation, it has disadvantages in certain scenarios.
- ----