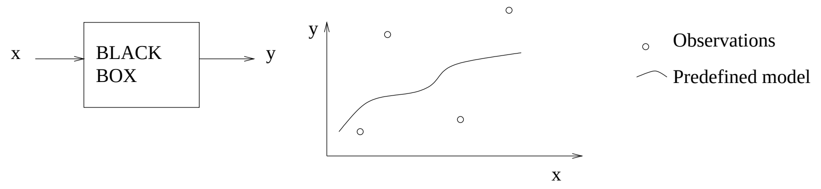
T-61.5040 Oppivat mallit ja menetelmät
T-61.5040 Learning Models and Methods
Pajunen, Viitaniemi

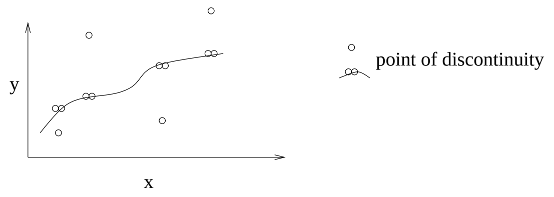
Solutions to exercises 1, 19.1.2007

Problem 1.

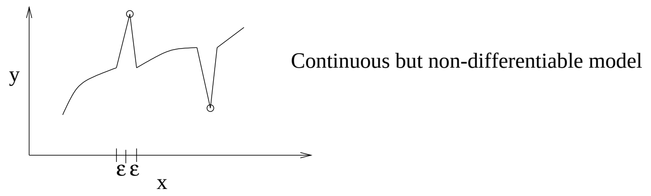
We show that in each case we can find a model that fits perfectly to the observations but yet it does not give any reasonable prediction outside the observations. The system is as follows:



i) We modify the predefined model only at the observations, so that the model gives a correct value at the observations.



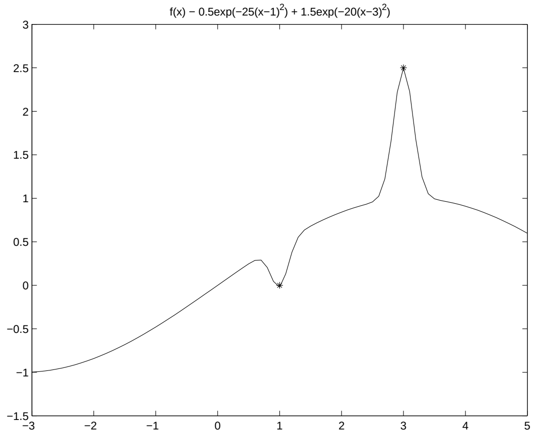
ii) We modify the predefined model in a neighborhood of radius ε of the observations. Here ε can be arbitrarily small. Now the model is continuos.



iii) Consider function $\exp(-x^2)$ that is infinitely many times differentiable. We can add a suitably scaled $\exp(-x^2)$ at the points of the observations. By scaling the argument of the exponential, $\exp(-x^2)$ can be made so thin that within a distance of ε from the observations, the values of the modified model and the original model differ only at most δ.
The value of the exponential must be scaled so that the value of the observations is "reached" (see figure), and its sign is chosen positive or negative according whether the value of the observation is larger or smaller than the value of the original model. Also, the exponential must be centered at x_i for observation i. The new model is thus

$$f(x) + \sum_i b_i \exp(-a_i(x - x_i)^2)$$

where f is the predefined model and a_i and b_i are scaling parameters, different for each observation. The figure shows an example of a situation where the original, predefined model (here, a sinusoid) is added with scaled exponentials at the observations.



Problem 2.

i) 8 bits per pixel and 256×256 pixels gives a total of 524 288 bits.

ii) Using a code of length n we can losslessly code 2^n different images, because each bit of the binary code can assume 2 values. Some of these images can be losslessly coded using a shorter code, too, for example using a code of length $(n-10)$. Using a code of length $(n-10)$ we can code 2^{n-10} different images. Thus among the 2^n original images, $\frac{2^{n-10}}{2^n} \approx 0.001$ or about one of a thousand images can be coded using this shorter code.

iii) Lossless compression of a natural image is not as difficult as described above. Natural images often contain redundancy: areas of constant gray level or repeating structures. Using this redundancy, the information content of the images can be presented in a compressed form. However, the compression will be succesful only if reasonably correct assumptions about the structure in images are made. Even if "generic" algorithms (such as LZW) may work fairly well, the assumptions are included (implicitly) in the algorithm.

Problem 3.

i) Any prediction will be equally good, since for any predicted bit there are exactly two vectors with the same $n-1$ bits and the last bits are different. The probability of making a mistake is 1/2 for any method.

ii) Assume a bag contains N vectors. Take any prediction method and compute its effectiveness by calculating the number of mistakes it makes for any given bag of vectors. You can construct another bag of vectors by flipping the last bit in each vector. All possible bags can now be paired so that the prediction method will make exactly N mistakes when applied to the pair of bags. We may assume that it is equally possible that the unknown bag is either one of this kind of pairs. Then the average error rate will be 1/2 for one vector. This is the same as achieved by guessing.

This shows that it is pointless to try to learn from data when there are no assumptions. An assumption would correspond to a specific bag, or a set of bags with a probability for each bag.

iii) We first assume that the training vectors are picked from the bag of vectors. Then we know the bag contains those vectors and possibly some unknown vectors. When a new vector is drawn, check if its first $n-1$ bits match any of the training vectors. If not, then guess the last bit since again you can make pairs of bags where the non-training vectors have their last bit flipped.

If the first $n-1$ bits match a training vector, then predict the last bit to be the same as for the training vector. You might use f_1 to do this. This follows because you have no a priori reason to assume that the bag contains an unbalanced set of unknown vectors with the same $n-1$ first bits.

If the training are not picked from the bag of vectors, any strategy performs as well as guessing (by part ii). The strategy outlined earlier is optimal also in this case. Thus we may conclude that the above mentioned strategy is optimal even though the problem statement leaves it unclear, whether the training vectors are picked from the vector bag.

We see that training data alone tells us something about the training data itself but nothing else. Again without assumptions, you cannot make better predictions than guessing outside the training data.

iv) One can again consider all possible bags containing 10% of the possible binary vectors and make pairs out of them by flipping the last bit. This will make the prediction error to be 1/2 on average.

This shows that even when the problem has "structure", as one might assume that real-world problems have, it is not possible to learn from data without making fairly correct assumptions about the structure.

Comments on all parts: these problems illustrate the ideas behind the No Free Lunch-theorems by David H. Wolpert (see e.g. Neural Computation 8, 1341-1390 (1996). The actual theorems are more general than predictions of bits but the ideas are more or less obvious from the above problems.)

Problem 4.

In a randomly chosen binary bag, the expected prediction error is 1/2 for the same reason as in the previous problem, part iv).

For the specific bag in which the BVM seems to perform so well the situation is a bit more delicate. The expected prediction error on vectors whose first $n-1$ bits do not match with any of the sampled n vectors is 1/2 for the same reason as in the previous problem, part iv). The vectors that have $n-1$ bits common with some sampled vector bias the expectation somewhat, but numerically the difference is small for any larger n as the proportion of matching vectors in the bag is at most $\frac{2n}{0{,}1\cdot 2^n} = \frac{5n}{2^{n-2}}$, e.g. for $n = 15$ less than 1/100.

The problem illustrates the "validation"of a learning method by simply testing it on some training data. If one really makes no assumptions about the contents of the bag, then the good performance of BVM on training data is just luck. Otherwise the BVM and the bag are "matched", i.e. the contents of the bag are such that BVM can be expected to work better than average.

Does the good performance on training data suggest that BVM and the bag of vectors "match"? It does not, using the argument from the previous problem, part iii). Training set without any additional information cannot give information about unobserved data. Therefore we cannot guarantee that the performance of BVM is good outside the training set.

This problem illustrates that the commonly used validation of a heuristic learning method is actually wrong, unless additional assumptions are made. Often these assumptions are not explicitly stated, but are implicit.

Problem 5.

First assume the algorithms A and B to be deterministic. We have a set of all binary bags $\{B_i\}$, B_i denoting the i:th bag. Let us write $C_A = C_A(B_i)$ for the proportion of prediction errors of the method A for the bag B_i. Correspondingly, $C_B = C_B(B_i)$ is the proportion of prediction errors of the method B for the bag B_i.

If we assume that any bag B_i is equally probable, then C_A and C_B are random variables. The difference $D = C_A - C_B$ is also a random variable. We are interested in the distribution of D, more specifically the tail probabilities of the distribution.

D can be written as a sum over the prediction errors on all vectors in the bag. Since any bag is equally probable, then each prediction error is independent from the others.

Write D informally as $\sum_j 2^{-n+1}(C_A(x_j) - C_B(x_j))$ where x_j is a vector in a bag. Each term is in the interval $[-2^{-n+1}, 2^{-n+1}]$. Then we can use the Hoeffding inequality to obtain

$$p(D - \mathrm{E}(D) \geq \epsilon) \leq \exp(-2\epsilon^2/2^{5-n})$$

From this we get easily

$$p(|D - \mathrm{E}(D)| \geq \epsilon) \leq 2\exp(-2\epsilon^2/2^{5-n})$$

(Where does the Hoeffding inequality come from? Plug the distribution of the given form to Chernoff bound $P(X \geq \lambda) \leq \inf_{r\geq 0} \mathrm{E}[e^{r(X-\lambda)}]$, choose suitable r and approximate the result.)

Above we assumed the algorithms A and B are deterministic. However, the results hold also for non-deterministic algorithms. This can be seen as follows: let r.v G denote all the non-determinism in the algorithms A and B. We can interchange the order of randomly selecting the bag and picking the value for G as the algorithms are independent of the chosen bag. If we first fix G, the remaining parts of the algorithms are deterministic and for a fixed value of G we can get the results as below. But the results then hold for any choice for G, i.e. always regardless of the non-determinism.

When we plug in the numbers given in the assignment, we obtain a limit for the tail probability:

$$P(|C_A - C_B| \geq 1/128) \leq 2\exp(-2\epsilon^2/2^{-15}) < 0.04$$

Interpretation: if you are predicting bits using 20-bit input vectors with any given algorithm, then in less than four percent of all such prediction problems is your performance to be even slightly different than that obtained by guessing. Thus beating guessing is not only hard on average, it also happens in a very small part of all problems (for deterministic algorithms).

We could alternatively use Chebychev's theorem to get a bound for the tail probabilities of the distribution of D:

$$\mathrm{P}(|D - \mathrm{E}(D)| > \epsilon) \leq \frac{1}{\epsilon^2}\mathrm{Var}(D).$$

We know that D gets values in the range $[-1, 1]$ and is symmetric about the origin. Therefore the expectation $\mathrm{E}(D) = 0$.

Let N denote number of vectors for which A and B disagree. Let S denote the number of cases where A is wrong and B is right and $T = N - S$ the opposite cases.

The number S of ones is has a $\mathrm{Bin}(N, 1/2)$ distribution. Hence

$$\mathrm{E}(S) = N/2$$

and

$$\mathrm{Var}(S) = N/4.$$

Now

$$\begin{aligned}\mathrm{Var}(D) &= \mathrm{Var}(2^{-(n-1)}[S - T]) = 2^{-2n+2}\mathrm{Var}(2[S - N/2]) = 2^{-2n+2} \cdot 4\mathrm{Var}(S) \\ &= 2^{-2n+2}N \leq 2^{-2n+2}2^{n-1} = 2^{-n+1}.\end{aligned}$$

Here we used $N \leq 2^{n-1}$. We are now ready to apply Chebyshev's inequality:

$$P(|D| \geq \epsilon) < \frac{1}{\epsilon^2}2^{-n+1}.$$

When we plug in the numbers given in the assignment, we obtain a limit for the tail probability:

$$P(|C_A - C_B| \geq 1/128) < 0.07.$$