Machine Learning (521289S)

Optimizing Classifier Performance

M.Sc. Antti Isosalo

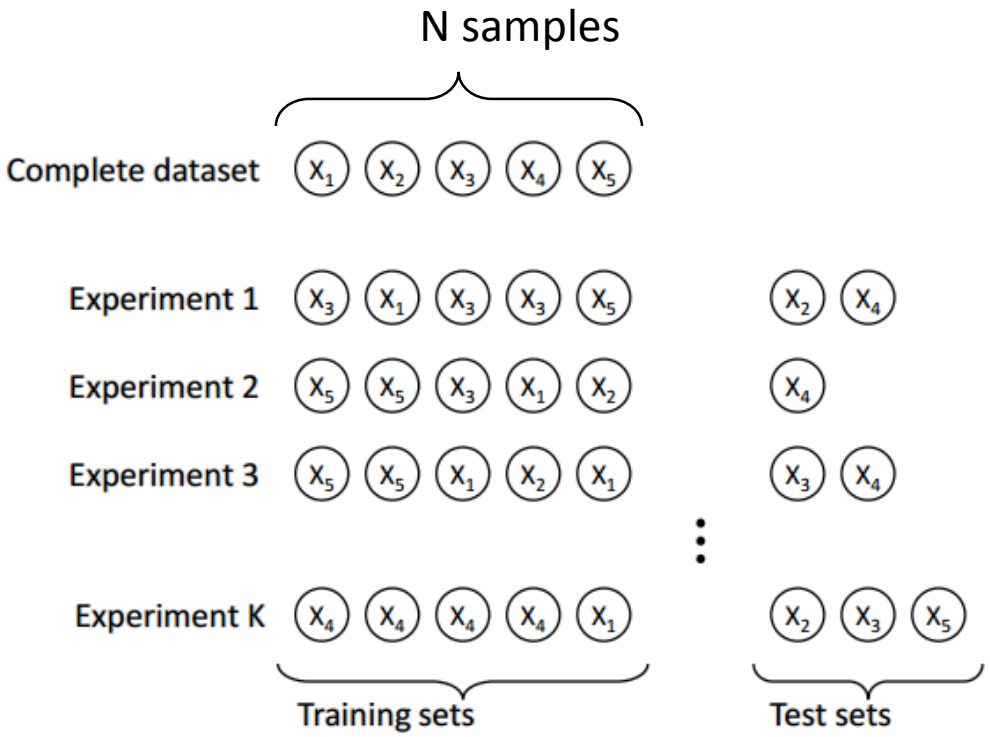
Physiological Signal Analysis Team
Center for Machine Vision and Signal Analysis (CMVS)
University of Oulu

Optimizing Classifier Performance

- Important part in making a well performing pattern recognition system is ***designing a set of features that produce separation*** between patterns.
  - In addition, we many times need methods which perform, for example, ***feature selection*** to find (most) optimal set of features.
- With good set of features we then have a chance to achieve good classification results.
  - If one algorithm seems to outperform another in particular situation, it is most likely results from the fact that it just fits well to that particular pattern recognition problem, not the general superiority of the algorithm.
- Also some ***assessment of the classifier performance*** must be made to be sure that the results we obtained are as good as we think.
  - For example ***cross validation*** to partition data into disjoint sets for separate learning and validation phases.
  - It is important that our classifier also generalizes to new unknown data which is the true measure of our chosen algorithm (cross validation helps to assess how poorly our model behaves on unseen data).

Bootstrapping

- ***Bootstrapping*** (or Bagging) uses multiple versions of a training set, each created by drawing $n' < n$ samples from D with replacement.
- Each of these bootstrap data sets is used to train a different ***component classifier*** and the final classification decision is based on the simple vote of each component classifier.
- Traditionally the component classifiers are of the same general form (i.e., all neural networks, or all decision trees)
- Only the final parameter values differ among them due to their different sets of training patterns.



New datasets are generated from the original set by sampling with replacement.

Boosting

- One approach in designing a good pattern recognition system is to ***design several*** different so-called ***weak learners*** (base learning algorithms) to address the same problem ***and then combine them*** as one final hybrid classifier.
- None of the weak learners themselves are perfect on their own.
- Combining them might though "boost" the classifier accuracy ***if their decisions are weighted in a proper way*** when combining.
- ***Boosting*** is one such method, where prediction rule is achieved by combining rough and moderately accurate classifiers.
- One of the most popular boosting methods is adaptive boosting, ***AdaBoost***.
  - AdaBoost concentrates on the difficult patterns.
- Any classifier, that can be taught recognize specific patterns, can be used here as weak learners (e.g., *k*-NN, Neural Network).

AdaBoost Algorithm For Two Class Problem With One Weak Learner

Samples: $\mathbf{x}_1, \ldots, \mathbf{x}_n$, data sample $\mathbf{x}_i \in X$

Outputs: $y_1, \ldots, y_n$, class label $y_i \in Y = \{-1, +1\}$

Initial weights: $D_1(i) = \frac{1}{n}$, $i = 1, \ldots, n$

$$sign(\sigma) = \begin{cases} 1 & \text{if } \sigma > 0 \\ 0 & \text{if } \sigma = 0 \\ -1 & \text{if } \sigma < 0 \end{cases}$$

Algorithm:

(a) Assemble the data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$

(b) Initialise weights $D_1(i)$ for the data samples

(c) For $t = 1, \ldots, T$

1. Train weak learner h using distribution D_t and compute weak hypothesis $h_t : X \to \{-1, +1\}$
2. Calculate the error (the sum of weights for which $h_t(\mathbf{x}_i) \neq y_i$)

$$\epsilon_t = \sum_{\substack{i=1 \\ h_t(\mathbf{x}_i) \neq y_i}}^{n} D_t(i)$$

of the classifier h_t

3. Calculate $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
4. Calculate $Z_t(i) = D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$
5. Update

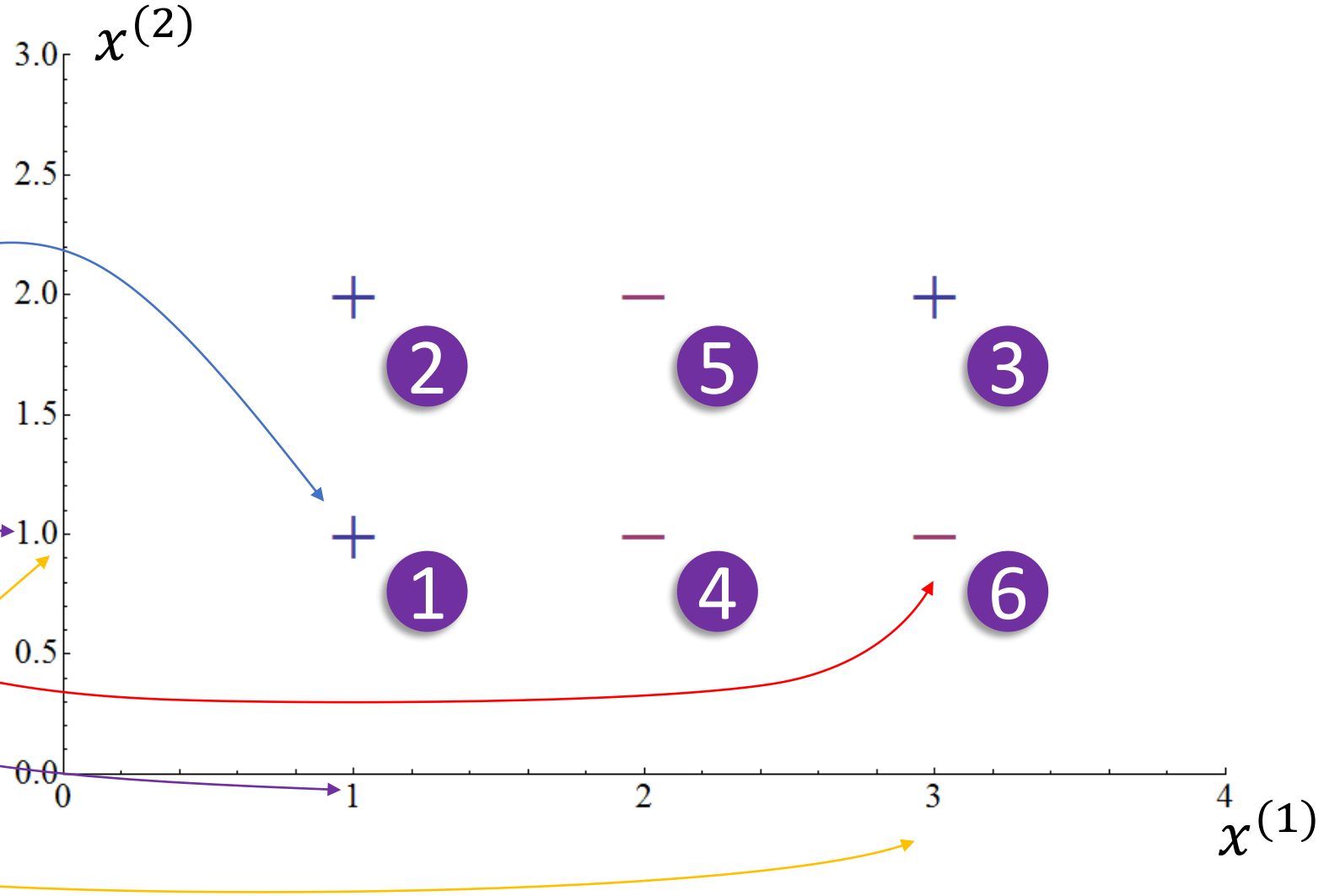
$$D_{t+1}(i) = \frac{D_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}}{\sum_{i=1}^{n} Z_t(i)},$$

where the normalization factor has been chosen so that D_{t+1} will result a distribution.

(d) Output the ready hybrid classifier (i.e., final hypothesis): $H(\mathbf{x}) = \text{sign} \left\{ \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right\}$

Exercise 1: AdaBoost

i	$x_i^{(1)}$	$x_i^{(2)}$	y_i
1	1	1	+1
2	1	2	+1
3	3	2	+1
4	2	1	-1
5	2	2	-1
6	3	1	-1



Exercise 1: AdaBoost (cont.)

- In the beginning, each of the data points have the same weight (see the algorithm).
- First, we must choose a linear classifier $h_1(x) = h_1(x^{(1)}; x^{(2)})$ so that sum of the weights ($D_1(i)$) of the misclassified points is as small as possible.