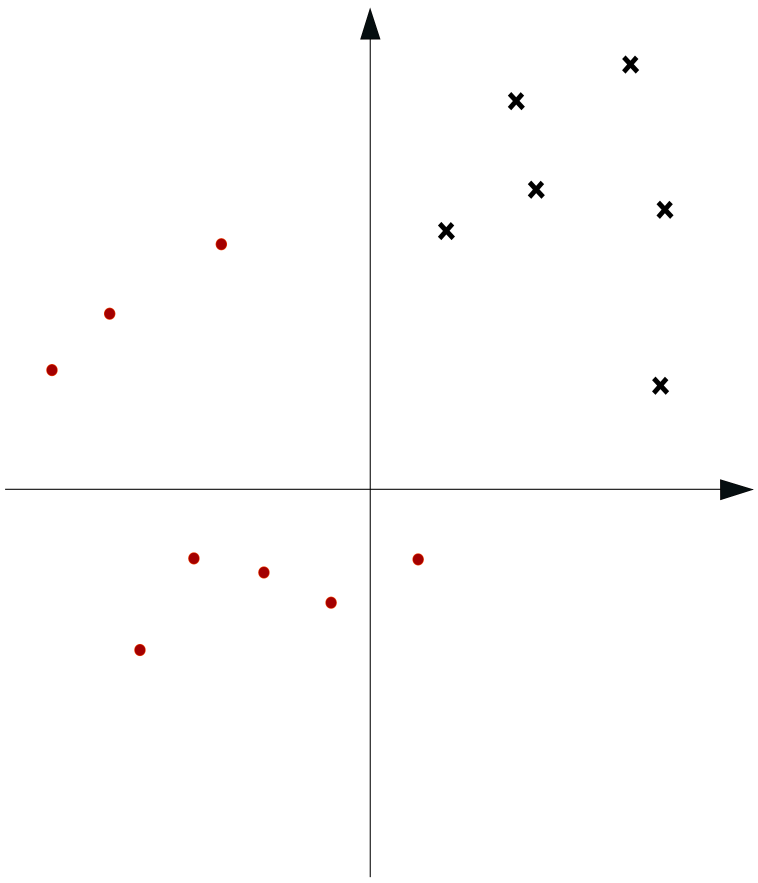
582744 Advanced Course in Machine Learning

Exercise 5 **Due April 25, 23:55 AM**

Rules:

1. Return your solutions in Moodle by the deadline.
2. The submission consists of two parts: (a) A single PDF file containing your answers to all questions. (b) A single file containing your code (either a single plain text source code or a compressed file). Do not include datasets, plots etc in this file, only the code.
3. If you feel comfortable, add an estimate of how many hours you worked on the problems in the beginning of your report.
4. Please typeset your work using appropriate software such as LaTeX. However, there is no need to typeset the pen and paper answers – you can also include a scanned hand-written version.
5. Pay attention to how you present the results. Be concise.
6. Spotted a mistake? Something is unclear? Ask for clarifications in Moodle.

This set of exercises is due on Tuesday April 25, before 23:55 AM.



1 Support vector machines (3 pts)

Answer the following questions related to SVMs.

(a) The figure above presents a two-dimensional data set with two classes, indicated by red dots (class -1) and black circles (class +1). Figure out the solution of a linear unregularized SVM classifier for this data set, by simply thinking about the properties of SVMs – no need to run any algorithms. Then mark in the figure the following things:

- The decision surface that separates the two classes
- The weight vector $\mathbf{w}$
- The support vectors corresponding to $\boldsymbol{\alpha}_i > 0$
- The margin of the classifier

Then draw two new samples into the data set so that the classes are no longer linearly separable (and assume the decision surface does not change – this would not be guaranteed in real case). Do this so that for one of the new samples the slack variable ϵ_i would be between 0 and 1 and for the other it would be bigger than 1.

(b) The lecture slides said that the augmented loss

$$L(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_i \boldsymbol{\alpha}_i \left[(\mathbf{w}^T\mathbf{x}_i + b)y_i - 1\right]$$

where $\boldsymbol{\alpha}_i$ are Lagrangian multipliers (and hence need to be non-negative) can be minimized by solving the optimization problem

$$\begin{aligned} \min \sum_i \boldsymbol{\alpha}_i - \frac{1}{2}\sum_{i,j} \boldsymbol{\alpha}_i \boldsymbol{\alpha}_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{s.t.} \quad \boldsymbol{\alpha}_i \geq 0 \quad \forall i \\ \sum_i \boldsymbol{\alpha}_i y_i = 0, \end{aligned}$$

where the optimization is now over $\boldsymbol{\alpha}$. Prove this by first solving for $\nabla_{\mathbf{w}} L(\mathbf{w}) = 0$ and $\nabla_b L(\mathbf{w}) = 0$, and then plugging the resulting expressions into the loss. The gradients were already provided on the lecture slides but it is a good idea to re-derive those as well.

(c) Explain briefly how the SVM problem can be kernelized.

2 Decision trees and forests (5 pts, programming)

In this exercise we implement a simplified decision tree and then build a random forest using the simplified trees. The tree we consider is a decision stump, a tree of exactly one decision node.

(a) First implement the decision stump so that:

- It picks a random input feature to be used for the decision
- It computes the information gain (of the class labels) for all possible thresholds for that input feature
- It picks the decision threshold with the highest information gain
- When used as a classifier, it outputs the probabilities for each class (that is, not just the most likely class)

The information gain is defined as the reduction in class entropy after making the decision. If the initial set of samples is denoted by D (with $|D|$ samples) and the decision rule splits it into two parts L and R, then the information gain for that rule is

$$H(Y \in D) - \frac{|L|}{|D|} H(Y \in L) - \frac{|R|}{|D|} H(Y \in R).$$

Here the notation $H(Y \in D)$ refers to estimating the entropy of the class variable Y based on the subset of samples that belong to the set D. To compute the entropy you need to count the number of samples for each class n_c, normalize them into a distribution $p(Y = c) = \frac{n_c}{\sum_k n_k}$, and then use the definition $H(Y) = -\sum_c p(Y = c) \log p(Y = c)$.

(b) Construct a random forest out of the decision stumps by simply learning M such stumps, using randomly selected subset of n samples for training each of them. The predictions of the random forest are then obtained by averaging the predictions over all of the M stumps. You can here use $n = 100$ samples for training each stump, unless you can think of a better choice.

(c) Apply the classifier on classifying the MNIST digits studied in Exercise 4. Train the model on the first 5000 samples and test on the 5000 samples following those. Note that some input features in MNIST are identically zero – it is a good idea to drop those before running the algorithm.

Plot the training and validation losses as a function of M (you should train for example $M = 2000$ models once and then just compute the errors for different sizes of ensembles, instead of every time learning a new set of stumps), using the classification error as the loss (error of one if the model gives highest probability for a class that is not the true one). Is the forest you created a good classifier? If not, can you think of how to improve it?

Hints:

- It is a good idea to represent the class labels as ten-dimensional binary vectors where 1 indicates the class label.
- You probably want to sort the inputs values for finding the decision threshold. Then you can compute the entropies for all possible thresholds in one loop by using cumulative counts of the class indicator vectors.
- Remember to have some safeguards for empty leaves – perhaps output a uniform distribution over the classes if that happens
- You do not need any actual data structures for trees here, since we only consider individual decisions