NUMERICAL OPTIMISATION
PROJECT GUIDELINES

Support vector machines (SVMs)

MARTA BETCKE, BOLIN PAN

Submission via Turnitin by 4pm on Tuesday 20th April 2020

Background research and problem formulation

Support vector machines (SVMs) are a well established and rigorously funded technique for solution of classification problems in machine learning. Your task is to research the mathematical problem setting for SMVs, write a short summary of your findings and propose *a plausible* simulation study culminating in an optimisation problem(s). In your report in particular consider:
Baseline:

- The mathematical derivation of the primal and dual problem for SVMs. The binary classification problem is considered a baseline. You are free to embellish by considering more classes, different penalty functions etc.
- Hard versus soft margin problem.
- Discuss the data set chosen. What are the challenges and how are you going to address them?

Challenge element examples:

- Considering more classes, different penalty functions etc.
- Nonlinear classification - kernels.
- Consider algorithms which are not on the syllabus.

Your study could be for instance based on considering different outlier patterns (kernels) or different loss functions etc.

Your study will yield an optimisation problem (or different problems in case of using different functions), consider what type of optimisation problem it is (convex / non-convex, constraint / un-constraint, smooth / non-smooth, linear / quadratic / non-linear, etc) and what are the challenges associated with it.

Solution of the optimisation problem

The goal here is to *perform the proposed simulation study* which will involve a solution of the resulting optimisation problem(s) using *two different numerical optimisation algorithms*. Set out the details of your study and classify the resulting optimisation problem(s). Briefly describe the two considered algorithms and recap the relevant theory such as convergence types and rates – relate the theory to your problem. Motivate your choice of optimisation algorithms. Compare the practical performance of the algorithms and relate it to their expected performance (the

theoretical convergence statements). If you compare the iteration counts between the methods, keep in mind the cost of an individual iteration, if these vary, compare the CPU times too. Provide explanation / rules for all parameter choices in your methods – choosing by inspection is perfectly valid but explain how you did it. Visualise your results, consider appropriate metrics of success and failure.

Examples of particular studies include (note that a study varies only one "aspect" of the problem):

- Consider two *different* loss functionals. Motivate your choices. Discuss their challenges and merits in optimisation context. Discuss the choices of algorithms for the different functionals (why is this algorithm a good choice?).
- Consider two different kernels for nonlinear classification problem.

Take care to design your study such that the resulting optimisation problems have sufficiently similar characteristics that they both can be solved by the two chosen algorithms.

Some pointers to related literature for SVMs. See also the references within.

- Andrew Ng, CS229 Lecture notes , *Support vector machines, Part V.*
- Tristan Fletcher, *Support Vector Machines Explained*

Extra drop-in session in week 9 or 10 (most likely in Tuesday lecture slot).

Marking

The project will be marked in line with UCL's project marking scheme: 70% for the standard component (very good treatment of the baseline) and 30% for the challenge component. The challenge level should be raised by considering *more involved setting not by increasing the number of studies.*

Your report should be a coherent and well structured document. It should contain to the point treatment of the topics. I do not specify the minimum number of words – the work will be evaluated on substance not length but succinctness is preferable – do not exceed 4000 words Please only include figures which you created yourself. Beware, that UCL rules on plagiarism apply.

Breaking down the mark for the project:

Determine the difficulty coefficient, $diff = 70\%$ + up to 30% for challenge
Each challenge element is worth 5-10% depending on difficulty. E.g. including kernels would be extra 10%.
The report: 80% which splits between:

- Content: 60%
- Structure & clarity: 20%
- Right level of mathematical rigour: 10%
- References: 10%

Code: 20%.

Mark for the code is almost binary. Sufficient code provided or not.

Final mark for the project will be determined as

$$diff * (content + structure + rigour + refs) + code.$$