Large Scale Learning

Sören Sonnenburg
Fraunhofer FIRST.IDA, Berlin

joint work with
Gunnar Rätsch, Christin Schäfer and Bernhard Schölkopf

Fraunhofer Institut
Rechnerarchitektur
und Softwaretechnik

Large Scale Problems

What makes a Problem Large Scale?

- millions of data points
- extreme high dimensionality
- high effort algorithms $\mathcal{O}(N^3)$
- large memory requirements

Applications

- Bioinformatics (Splice Sites, Gene Boundaries,...)
- IT-Security (Network traffic)
- Text-Classification (Spam vs. Non-Spam)

Creates Challenges just to load/process data.

Are LS methods necessary?

Needs methods that scale linearly $\mathcal{O}(N)$

Conjectures:

- higher effort (e.g. $\mathcal{O}(N^2)$) algorithms may be superior over "simpler" $\mathcal{O}(N)$ ones
- large scale methods are not necessary - lets just sub-sample the data
- approximations vs. exact algorithms

There are not many LS problem types

Conjectures:

- all large scale learning problems are either very sparse and high dimensional or come with millions of data points

Many people claim they have a faster algorithm, but hey in fact have not

Conjectures:

- all large scale learning problems are either very sparse and high dimensional or come with millions of data points
- many methods are faster only because they don't solve the true (same) problem or have a different stopping criterion
  - Example: Linear SVMs, SVM^{light}, SVM^{perf}, Pegasos, BMRM-SVM, SVMLin
  - some SVMs are not optimized to the very end but some potentially arbitrarily far away solution is found, different objective

Does that matter in practice?

- What is the true goal? Classification Accuracy?!
- How can one setup a fair comparison?
- How could one organize a challenge on LS learning?
  - Do you have large (diverse) datasets?
  - How can one judge whether algorithm X is faster than algorithm Y?
  - Some Methods are just faster as they use assembly language to in their core loop (e.g. HeroSVM)
  - **For SVMs:** compare objective and training time.