BMC Bioinformatics

SOFTWARE Open Access

SparSNP: Fast and memory-efficient analysis of all SNPs for phenotype prediction

Gad Abraham[1*], Adam Kowalczyk[1], Justin Zobel[1] and Michael Inouye[2,3]

Abstract

Background: A central goal of genomics is to predict phenotypic variation from genetic variation. Fitting predictive models to genome-wide and whole genome single nucleotide polymorphism (SNP) profiles allows us to estimate the predictive power of the SNPs and potentially develop diagnostic models for disease. However, many current datasets cannot be analysed with standard tools due to their large size.

Results: We introduce SparSNP, a tool for fitting lasso linear models for massive SNP datasets quickly and with very low memory requirements. In analysis on a large celiac disease case/control dataset, we show that SparSNP runs substantially faster than four other state-of-the-art tools for fitting large scale penalised models. SparSNP was one of only two tools that could successfully fit models to the entire celiac disease dataset, and it did so with superior performance. Compared with the other tools, the models generated by SparSNP had better than or equal to predictive performance in cross-validation.

Conclusions: Genomic datasets are rapidly increasing in size, rendering existing approaches to model fitting impractical due to their prohibitive time or memory requirements. This study shows that SparSNP is an essential addition to the genomic analysis toolkit.
SparSNP is available at http://www.genomics.csse.unimelb.edu.au/SparSNP

Background

One of the challenges raised by recent advances in the genomics of complex phenotypes is the prediction of phenotype given genotype, such as prediction of disease from SNP data. Successful identification of SNPs strongly predictive of disease promises a better understanding of the biological mechanisms underlying the disease, and has the potential to lead to early disease diagnosis and preventative strategies. The question of predictive ability is also closely related to the proportion of phenotypic and genetic variance that can be explained by common SNPs and the lively debate surrounding the "missing heritability" of many complex diseases [1]. To quantify the genetic effect, we must fit a statistical model to all SNPs simultaneously. Lasso-penalised models [2] are well suited to this task, since they perform variable selection — some model weights are exactly zero and thus excluded from the model. In this way, lasso models remove the need for screening SNPs based on univariable statistics prior to fitting a multivariable model [3].

However, fitting models to genome-wide or whole-genome data is challenging since such studies typically assay thousands to tens of thousands of samples and hundreds of thousands to millions of SNPs. With standard analysis tools, modelling genome-wide and whole genome data is either impossible or extremely inefficient. For example, most existing analysis tools require loading the entire dataset into memory prior to fitting the models, which is both time-consuming and requires large amounts of memory to store the data and fit the models. In order to perform simultaneous modelling of SNP variation across the genome and build predictive models of disease and phenotype, it is clear that there is a need for new tools that are fast, not memory intensive, and easy to use.

Here, we present the tool SparSNP, which is an efficient implementation of lasso-penalised linear models. SparSNP can fit lasso models to large-scale genomic datasets in minutes using small amounts of memory, outperforming equivalent in-memory methods. Thus,

*Correspondence: gabraham@csse.unimelb.edu.au
[1]NICTA Victoria Research Lab, Department of Computing and Information Systems, The University of Melbourne, Parkville 3010, Victoria, Australia
Full list of author information is available at the end of the article

SparSNP makes it practical to analyse massive datasets without the use of specialised computing hardware or cloud computing. SparSNP produces cross-validated model weights that can be used to select the top predictive SNPs. SparSNP also allows the resulting models to be evaluated for predictive power and phenotypic/genetic variance explained. The main features of SparSNP are:

- implementation of ℓ_1-penalised linear regression for continuous traits and ℓ_1-penalised classification for binary traits;
- speed — SparSNP fits models to data with 10^4 samples and 5×10^5 SNPs in < 10 minutes on standard hardware;
- small (and tunable) amounts of memory are required: $\sim$ 1GiB for the datasets analysed here;
- compatibility with PLINK [4] BED (SNP-major ordering) and FAM files (single phenotype);
- cross-validation is performed natively, removing the need to manually split datasets;
- convenient external validation if an independent dataset is available;
- conveniently produces a set of models with increasing numbers of SNPs in each model, allowing for model selection based on cross-validated predictive performance;
- calculates the area under receiver-operating-characteristic curves (AUC) and explained phenotypic or genetic variance, in cross-validation or on validation datasets.

An outline of the SparSNP analysis pipeline is shown in Figure 1. See Additional File 1 for details of the analysis workflow.

Results

To assess the performance of SparSNP and compare it with existing methods, we used a celiac disease case/control dataset [5], consisting of N = 11,940 samples from five European populations (Italian, Finnish, two British, and Dutch), with p = 516,504 autosomal SNPs. The data processing and quality control have been described in the original publication.

We performed two types of comparisons, one for computing speed and one for predictive performance. For the speed comparison, we timed the process of fitting the model to data subsets of different sizes. For the predictive comparison, we used cross-validation for one population

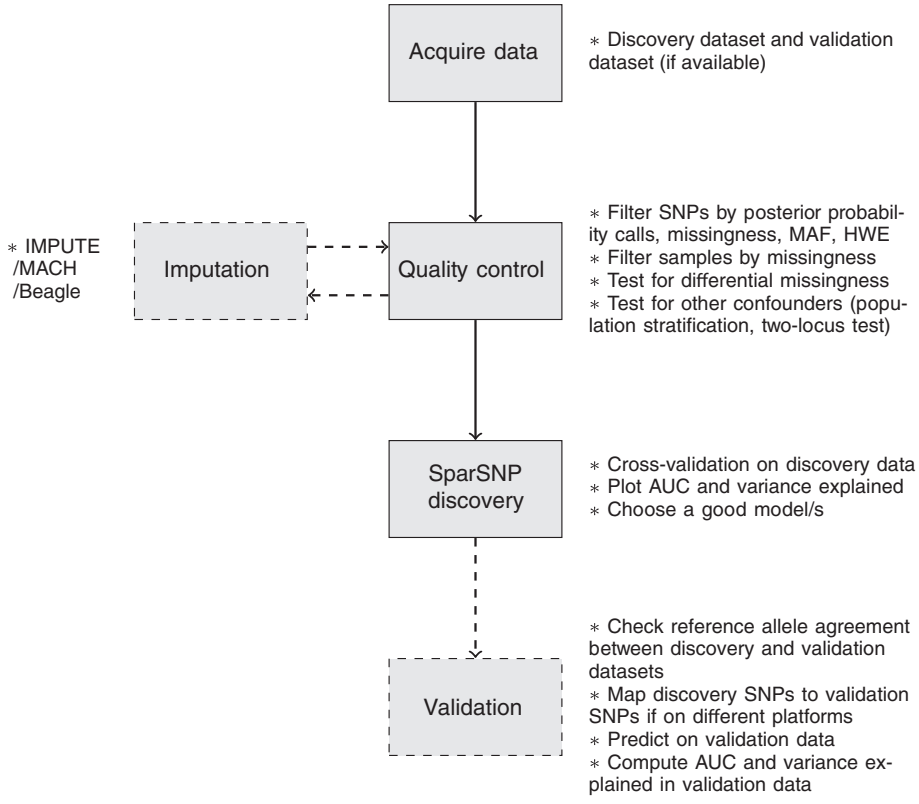

Figure 1 SparSNP analysis pipeline. An example pipeline for analysing a SNP discovery dataset with SparSNP and testing the model on a validation dataset. Most of the data preparation and processing can be done with PLINK.

(Finns) over a grid of hyperparameters. We compared the following methods:

- SparSNP 0.89[a], with ℓ_1-penalised squared hinge loss, implemented in C(see Additional Files 3 and 4);
- glmnet 1.7 [6][b], with logistic loss (binomial family), implemented as a Fortran library for R [7]. glmnet implements two variants of cyclical coordinate descent, together with warm restarts and active set convergence;
- LIBLINEAR 1.8 [8][c], with ℓ_1-penalised squared hinge loss (model 5), implemented in C++. LIBLINEAR uses coordinate descent to optimise the loss function;
- LIBLINEAR-CDBLOCK [9][d] with block ℓ_2-regularised squared hinge loss (model 1), implemented in C++. LIBLINEAR-CDBLOCK splits the data into blocks of user defined size, loads each one into memory, and performs a block-wise optimisation of the loss function;
- and HyperLasso [10][e], logistic regression with the double exponential (DE) prior (equivalent to lasso), implemented in C++. HyperLasso implements cyclical coordinate descent as well.

All five methods assumed a model that is additive in the minor allele dosage {0, 1, 2}.

Table 1 summarises the relative strengths and weaknesses of the five methods evaluated here, with respect to memory requirements, speed, best AUC in prediction, whether the tool successfully fitted a model to the largest dataset, the number of genetic models available, and ease of use. We now present these results in detail.

SparSNP makes possible rapid, low-memory analysis of massive SNP datasets

SparSNP consistently outperformed the other methods when fitting models (Figure 2). We ran all methods on random subsets of the celiac disease dataset, consisting of randomly selected subsets of the data with p = {50,000, 250,000, 500,000} SNPs and N = {1000, 5000, 10,000} samples, a total of nine subsets. This process was independently repeated 10 times. Only SparSNP and LIBLINEAR-CDBLOCK could fit models to datasets with $>$ 5000 samples and 250,000 SNPs, and they were the only tools that could fit models to $>$ 1000 samples and 500,000 SNPs on a machine with 32GiB RAM. It is important to note that the aforementioned data sizes would be considered quite small by current standards. Also note that in contrast with SparSNP, LIBLINEAR-CDBLOCK does not implement an ℓ_1-penalised model but a standard ℓ_2-penalised support vector machine (SVM), which is not a sparse model, and does not produce solutions over a grid of model sizes; instead, a computationally expensive scheme such as recursive feature elimination (RFE) [12] would be required in order to find sparse models, but we did not use RFE here. Of the remaining methods, LIBLINEAR and glmnet did not complete all experiments due to running out of memory (on a 32GiB RAM machine) or due to the data exceeding the limit on matrix sizes in R (a maximum of $2^{31} - 1$ elements). HyperLasso took much longer to complete: $\sim$ 2 hours for the 1000 sample/500,000 SNP subset and $\sim$ 69 hours for the 10,000 sample/500,000 SNP subset. Therefore, the timing for HyperLasso is not shown.

We emphasise that these results are for one run over the data — in practice, cross-validation is used to guide model selection and evaluate the generalisation error of a model. Run times for cross-validation would be higher yet — 3-fold cross-validation repeated 10 times would take approximately 20 times longer, $\sim$ 22 and $\sim$ 4 hours for LIBLINEAR-CDBLOCK and SparSNP, respectively, over the largest subset (with SparSNP also outperforming LIBLINEAR-CDBLOCK in terms of prediction as we show in the next section) — making the differences in

Table 1 Comparison of the evaluated methods

Method	Memory required	Speed (rank)	Prediction AUC (rank)	Fitted largest data	Genetic models	Ease of use
SparSNP	●●●	(1)	(1)	yes	●○○○	●●●●●
glmnet	●●○	(2)	(1)	no	●○○○	●●●○○
HyperLasso	●●●	(5)	(3)	yes	●●●○	○○○○○
LIBLINEAR	●●○	(3)	(2)	no	●○○○	●●○○○
LIBLINEAR-CDBLOCK	●●●	(4)	(4)	yes	●○○○	●●○○○

We evaluated each method in terms of the following criteria:

[a] Memory requirements: maximum GiB required to complete the prediction experiment. Three points: $\leq$ 4GiB, as is commonly available on laptops. Two points: $>$ 4GiB and $\leq$ 32GiB, commonly available on compute servers. One point: $>$ 32GiB, typically available on higher-end servers.

[b] Speed: time to complete in the timing experiments with 50,000 SNPs (Figure 2).

[c] Prediction: best cross-validated AUC in the prediction experiment (Figure 3).

[d] Fitted largest data: whether the tool successfully completed the largest timing experiment, consisting of p = 500,000 SNPs and N = 10,000 samples.

[e] Models: one point for each natively supported model of (i) additive, (ii) dominant/recessive, (iii) heterozygous models, (iv) and interaction models.

[f] Ease of use: one point for each of (i) does the tool support input in formats commonly used in the genetics community, such as PLINK BED or PED files, (ii) does the tool implement cross-validation, (iii) does the tool estimate the AUC, R^2, or explained variance from the cross-validation, (iv) does the tool produce plots of the resulting AUC, R^2, or explained variance, for easy model selection and evaluation, and (v) does the tool implement native imputation of missing genotypes.

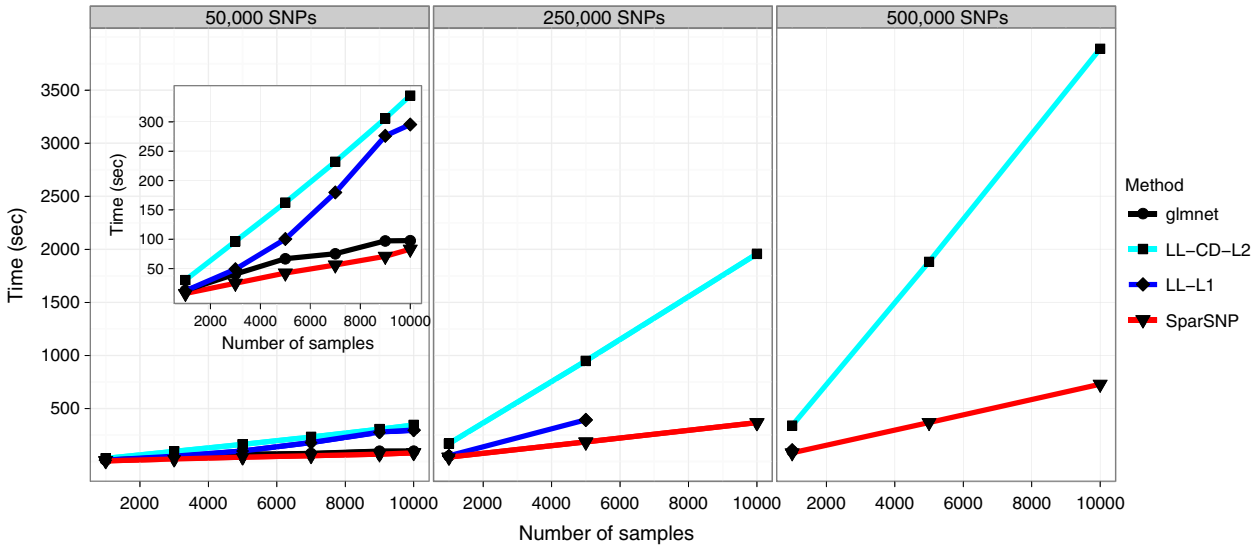

Figure 2 Timing experiments. Time (in seconds) for model fitting, over sub-samples of the celiac disease dataset, taken as the minimum time over 10 independent runs. The inset panel shows the results for 50,000 SNPs in more detail, note the different scales. For in-memory methods we included the time to read the data into memory. For SparSNP and glmnet we used a penalty grid of size 20, and a maximum model size of 2048 SNPs. LIBLINEAR (denoted "LL-L1") and LIBLINEAR-CDBLOCK (denoted "LL-CD-L2") induced one model with $C = 1$. LIBLINEAR-CDBLOCK used $m = 50$ blocks. For some datasets, glmnet and LIBLINEAR did not complete and these running times are not shown. HyperLasso is not shown since it took much longer to complete than the other methods.

speed even more important. Also note the difference in the number of models fitted: both SparSNP and glmnet use a warm restart strategy, computing a separate model for each penalty in a grid (we used 20 penalties), resulting in a path of 20 separate models with different sizes, whereas LIBLINEAR, LIBLINEAR-CDBLOCK, and HyperLasso computed only one model based on one penalty.

SparSNP produces models of better or comparable predictive ability

We used the Finnish subset of the celiac disease dataset ($N = 2476$ samples, $p = 516{,}504$ SNPs) to evaluate predictive performance of the models in 3-fold cross-validation. We measured predictive ability with the area under the receiver operating characteristic curve (AUC) [13], where AUC ranges from 0 (perfectly wrong prediction) to 1 (perfect prediction), with AUC = 0.5 being equivalent to random prediction (no predictive power). AUC also has the probabilistic interpretation as the probability of correctly ranking the risk of the cases higher than the risk for the controls, for a randomly selected pair of cases and controls. From the AUC we also estimated the explained proportion of phenotypic variance [11], assuming a population prevalence for celiac disease of $K = 1\%$. We did not evaluate the predictive ability over the entire celiac dataset, as it consists of several populations of different ethnic background, and case/control status may be confounded by effects such as population stratification.

SparSNP induced models with AUC of up to 0.9 and explained phenotypic variance of up to $\sim 40\%$ (Figure 3), almost identical to that of glmnet, except for small differences at the extremes of the λ path; the differences may be due to the fact that SparSNP and glmnet use different loss functions and have different parameters such as convergence tolerances. LIBLINEAR showed maximum AUC similar to glmnet and SparSNP, but much lower AUC for smaller number of SNPs in the model. LIBLINEAR-CDBLOCK showed consistently lower AUC over the range of costs used: a grid of 30 costs $C \in [10^{-4}, 10^{3}]$. Varying the costs did not substantially change the AUC. Since LIBLINEAR-CDBLOCK used an ℓ_2-SVM, which does not induce sparse models and does not natively produce a range of model sizes, we show results for a model with all 516,504 SNPs, averaged over all penalties.

Due to the high computational cost of running HyperLasso, we were not able to run as comprehensive a grid search; therefore, we performed only two replications of 3-fold cross-validation, using the DE prior with parameter $\lambda = \{2, 4, 8, 10, 12, 14, 16, 18, 20\}$ over 10 iterations (10 posterior modes), and averaged the AUC over the modes.

Importantly, while SparSNP achieved AUC better than or comparable to the other approaches, the resources consumed were far from being equal — SparSNP performed 3-fold cross-validation using a total of about 1 GiB of RAM, whereas LIBLINEAR required about 24GiB, and glmnet used up to 27GiB (the total number of samples used in the cross-validation training phase is $\sim$ 1650,

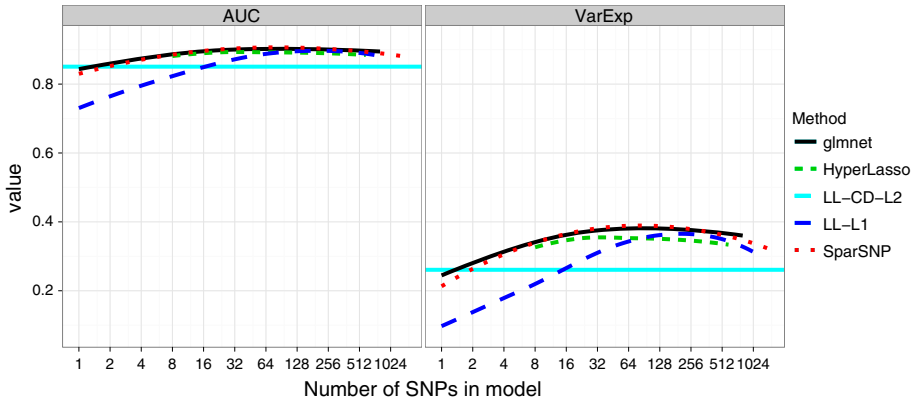

Figure 3 Prediction experiments. LOESS-smoothed AUC and explained phenotypic variance (denoted "VarExp"), for the Finnish celiac disease dataset, for increasing model sizes. AUC is estimated over 20 × 3-fold cross-validation, except for HyperLasso for which we ran only 2 × 3-fold cross-validation due to the high computational cost. The explained phenotypic variance is estimated from the AUC using the method of [11], assuming a population prevalence of celiac disease $K = 1\%$. Note that glmnet, HyperLasso, LIBLINEAR (denoted "LL-L1"), and SparSNP used an ℓ_1-penalised model, whereas LIBLINEAR-CDBLOCK (denoted "LL-CD-L2") used an ℓ_2-penalised model (non sparse), inducing a model using all 516,504 SNPs, therefore it is shown as a horizontal line across all model sizes. Note that tuning the ℓ_2 penalty for LIBLINEAR-CDBLOCK resulted in very similar AUC.

or 2/3 of the total Finnish subset). Both LIBLINEAR-CDBLOCK and HyperLasso used low amounts of memory: LIBLINEAR-CDBLOCK used about 210MiB of RAM (using 50 disk-based blocks), and HyperLasso used a maximum of only 2GiB (roughly the size of the training data), however, it was by far the slowest.

We also evaluated how the SNPs found by each method agreed with each other, when tuned to select 128 SNPs on the entire Finnish dataset (Figure 1 in Additional File 2), Overall, the sparse methods had high correlations of their SNP weights (≥ 0.86), with lower correlations for LIBLINEAR-CDBLOCK (0.21–0.23). SparSNP and glmnet shared a high proportion of the 128 non-zero weight SNPs (0.71), whereas LIBLINEAR-L1 and HyperLasso shared lower proportions of their selected SNPs (0.23–0.43), and LIBLINEAR-CDBLOCK shared the lowest proportions with the others (0.09–0.12). The differences in the selected SNPs are likely due to differences in the loss functions, differences in the penalisation (ℓ_1 versus ℓ_2), the numerical properties of each optimisation method, and high LD, as the lasso tends to select one SNP of out a group of highly correlated SNPs.

Discussion

Many genetic datasets have assayed thousands of samples over hundreds of thousands of SNPs. Currently, sample collections are expanding in multiple ways: by increasing sample numbers, by imputing millions of SNPs with fine-scale reference panels, and by performing whole-genome sequencing. There is thus a pressing need for analytical tools which are capable of handling such massive amounts of data. Here, we have presented SparSNP, a tool to rapidly perform phenotype prediction and variance estimation from massive amounts of SNP data.

The main bottleneck in the analysis is the large amounts of RAM required to fit models, which may not be feasible or accessible to many users. SparSNP incorporates multiple computational strategies to minimise the amount of RAM required. Even when such memory is available, the time taken to read the data from disk becomes the bottleneck, rather than the fitting process itself. Thus, the time taken to analyse the data may be long enough to preclude a comprehensive analysis of the data, such as multiple rounds of cross-validation or experimenting with various model parameters. In contrast, SparSNP makes it possible to rapidly analyse such datasets — 10 replications of 3-fold cross-validation of a 10,000-sample/500,000 SNP dataset can be performed in about 2 hours, requiring only ∼ 1GiB RAM. This time can be further reduced by running multiple instances in parallel on a compute cluster. While the celiac disease dataset analysed here is quite large, recent genome-wide studies are larger still, involving 1–6 million SNPs, either by direct assay or by imputation from HapMap [14,15] or 1000Genomes [16]. The number of samples in current datasets is larger as well, and likely to continue growing into the hundreds of thousands. For such studies, fitting multivariable models using current methods is not feasible with standard tools. SparSNP is scalable in terms of memory requirements, and yet is faster than comparable approaches, making it suitable for analysing such datasets. Further, SparSNP achieved similar or better predictive ability than other approaches. Our future work will include adding the ability to include external variables such as age and sex and other clinical phenotypes in the model, having multiple genetic models

(additive, dominant/recessive, heterozygous), the ability to model multiple phenotypes, and potentially implementing different loss functions such as Cox survival models.

Implementation

SparSNP implements an efficient out-of-core version of the cyclical coordinate descent method [6,17] for minimising ℓ_1-penalised loss functions. Here, we briefly discuss the main steps in th fitting process. See Additional File 2 for the details of the computational procedure.

ℓ_1-penalised loss functions

The problem of fitting linear models can be cast as minimising a convex loss function L. The squared loss function over N samples in p variables is used for linear regression and is defined as

$$L(\beta_0, \beta) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|, \qquad (1)$$

where $x_i \in \mathbb{R}^p$ are the inputs for the ith sample, $y \in \mathbb{R}^N$ is the N-vector of outputs, $\beta_0 \in \mathbb{R}$ is the intercept, $\beta \in \mathbb{R}^p$ is a p-vector of model weights. and $\lambda \geq$ is the user-chosen penalty. Another loss function useful in classification is squared-hinge loss, which is equivalent to a least-squares support vector machine with a linear kernel [18], and defined as

$$L(\beta_0, \beta) = \frac{1}{2}\sum_{i=1}^{N}\max\{0, 1-y_i(\beta_0+x_i^T\beta)\}^2+\lambda\sum_{j=1}^{p}|\beta_j|, \qquad (2)$$

where $y_i \in \{-1, +1\}$. SparSNP uses the squared hinge loss as the classification model for case/control data.

Out-of-core coordinate descent

ℓ_1 regression is a convex optimisation problem. However, in general, it has no analytical solutions, and must be solved numerically. We use a variant of coordinate descent to numerically minimise the loss function.

In coordinate descent [6,17,19], each variable is optimised with respect to the loss function using a univariable Newton step, while holding the other variables fixed. Since the updates are univariable, computation of the first and second derivatives is fast and simple (we assume that all our loss functions are twice-differentiable). The ℓ_1 penalisation is achieved using *soft thresholding* [17] of each estimated weight $\hat{\beta}_j$

$$\hat{\beta}_j \leftarrow S\left(\hat{\beta}_j - s_j, \lambda\right), \qquad (3)$$

where $s_j = \frac{\partial L}{\partial \beta_j} / \frac{\partial^2 L}{\partial \beta_j^2}$ is the Newton step with respect to β_j and $S(\cdot, \cdot)$ is the soft thresholding operator

$$S(\alpha, \gamma) = \text{sign}(\alpha)\max\{0, |\alpha| - \gamma\}, \quad \gamma \geq 0.$$

Model selection

The λ penalty tunes the model complexity, and can be selected in several ways. The simplest way is to leave it fixed at some arbitrary value, however, this may result in suboptimal performance if the number of selected variables is too small or too large. A second way is to pre-specify the number of non-zero SNPs required, and then perform binary search for the λ penalty that produces the required number of SNPs [3]; SparSNP does not support this option. The third and recommended way is to use cross-validation, as implemented in SparSNP, and choose a model or set of models that maximise the cross-validated AUC. These models can then be tested on an independent validation dataset to get unbiased estimates of AUC. We have found cross-validation to work well for genetic datasets of moderate to large size such as those used here, such that the training and test subsets are large enough and the case/control phenotypes are roughly balanced in the data. For highly imbalanced classes, other schemes such as stratified cross-validation may be required; these are currently not implemented in SparSNP.

Experimental setup

We employed two experimental setups, one for timing comparisons and another for comparisons of predictive ability.

Timing experiments

For input, SparSNP used the PLINK BED/FAM files, glmnet read unpacked binary versions of the BED files into R (one genotype per byte), LIBLINEAR read text files in LIBSVM sparse format, and HyperLasso read text files in its requisite text format. The time taken to convert the PLINK BED files into the appropriate formats was considerable, but not included in the timings reported here. For SparSNP, the timings include the time to scale the genotypes to zero-mean/unit-variance, fit the model, and write the model weights to disk. For glmnet, LIBLINEAR, and HyperLasso, the timings include the time to read the genotypes from disk, fit the model, and write the model to disk where appropriate. For LIBLINEAR-CDBLOCK, timing included the time to split the data into blocks and fit the model to the data. Since HyperLasso was considerably slower than the other methods, we only ran two simulations with it. The largest models allowed was 2048 non-zero variables (excluding the intercept). For LIBLINEAR, we used one default cost $C = 1$. LIBLINEAR-CDBLOCK used $m = 50$ blocks with warm restarts. For

HyperLasso, we used the double exponential (DE) with hyperparameter $\lambda = 1$, for one iteration.

We ran all timing experiments on a 2.6Ghz dual-CPU dual-core AMD Opteron 2218 machine with 32GiB RAM, running 64-bit Ubuntu Linux 8.04.4 with local disk drives.

Prediction experiments

For SparSNP and glmnet we used a grid of 20 decreasing lasso penalties λ, such that decreasing penalties induce models with more non-zero variables up to the maximum allowed number of 1024. For LIBLINEAR we used a grid of 30 costs $C \in [10^{-4}, 10^{3}]$, since the penalty is expressed as the cost $C = 1/\lambda$. Inputs for SparSNP and glmnet were scaled to zero mean and unit variance. For LIBLINEAR and LIBLINEAR-CDBLOCK, inputs were scaled to the range [-1, +1] using `svm-scale` from LIBSVM.

We ran all prediction experiments on an Intel Xeon X5550 2.67Ghz machine with 48GiB RAM, running Red Hat Enterprise Linux Server 5.6, with networked disk drives.

Data preparation

As with any association method, genotype-phenotype associations that are in the data but not truly of biological origin, such as batch effects, may confound the analysis, potentially resulting in the detection of spurious associations and inflation of the apparent predictive ability. The use of an independent validation dataset is highly recommended.

Missing data

For convenience, SparSNP implements simple random imputation for missing genotypes, where missing genotypes are randomly replaced with a genotype {0, 1, 2} (with probability 1/3 each). When the proportion of missingness is small and the genotypes are missing at random (for example, no differential missingness between cases and control), such a simple approach does not substantially affect the predictive ability and does not introduce significant spurious associations. However, when missingness is high or differentiated between cases and controls, spurious associations can arise and we recommend either using PLINK to filter SNPs and samples with high missingness, or alternatively, imputing the missing data using a more sophisticated method such as Beagle [20], IMPUTE [21], or MACH [22].

Confounding effects

SparSNP does not account for possible batch effects, which must be accounted for at the quality control stage. Nor does SparSNP currently account for confounders such as population stratification, admixing, or cryptic relatedness; EIGENSTRAT [23] and PLINK can be used to detect these and to filter the data accordingly.

Genetic Models

SparSNP implements models additive in the minor allele dosage {0, 1, 2}. Other models, such as dominant/recessive models or interaction models are currently not supported.

Applying models to new data

SparSNP produces text files containing the model weights for each SNP, and can be used in prediction mode to read these weights, together with another BED file, to produce predictions for other datasets. Model weights are with respect to the minor allele dosage for the training data, and the reference allele may be different in another dataset, possibly resulting in reversal in the sign of the SNP effect. In addition, both the discovery and validation datasets must contain the same SNPs in the same ordering (marker names are not important). We recommend using PLINK to ensure that both the discovery and validation datasets contain the same SNPs and are encoded using the same reference alleles.

Availability and requirements

Project name: SparSNP
Project home page: http://www.genomics.csse.unimelb.edu.au/SparSNP
Operating system(s): 64-bit Linux and Mac OS X
Programming language: NA
Other requirements: Bash, R
License: binaries only, redistribution is allowed, see website.
Any restrictions to use by non-academics: no restrictions

Endnotes

[a]http://www.genomics.csse.unimelb.edu.au/SparSNP
[b]http://cran.r-project.org/web/packages/glmnet
[c]http://www.csie.ntu.edu.tw/~cjlin/liblinear
[d]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/cdblock
[e]http://www.ebi.ac.uk/projects/BARGEN/download/HyperLasso

Additional files

Additional file 1: An example of a SparSNP workflow, covering basic quality control, training the model on discovery data, applying the model to validation data, plotting the results, and post-processing.

Additional file 2: Details of the implementation of SparSNP and other supplementary results.

Additional file 3: SparSNP v0.89, statically linked version for 64-bit Linux x86-64.

Additional file 4: SparSNP v0.89, version for 64-bit Mac OS X.

Abbreviations

SNP, Single nucleotide polymorphism; GiB, Gibibyte (2^{30} bytes); AUC, Area under receiver operating characteristic curve; I/O, Input/output; RAM, Random access memory.

Competing Interests
The authors declare that they have no competing interests.

Acknowledgements
Thanks to David van Heel (QMUL) for supplying the celiac disease data, and to the Victorian Life Sciences Computing Initiative (VLSCI) for providing computing facilities under project VR0126. Funding: MI was supported by an NHMRC Postdoctoral Fellowship (no. 637400). This work was supported by the Australian Research Council, and by the NICTA Victorian Research Laboratory. NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications, and the Digital Economy, and the Australian Research Council through the ICT Centre of Excellence program. This work was made possible through Victorian State Government Operational Infrastructure Support and Australian Government NHMRC IRIIS.

Author details
[1]NICTA Victoria Research Lab, Department of Computing and Information Systems, The University of Melbourne, Parkville 3010, Victoria, Australia. [2]Immunology Division, The Walter and Eliza Hall Institute of Medical Research, Parkville 3052, Victoria, Australia. [3]Departments of Pathology and of Microbiology & Immunology, The University of Melbourne, Parkville 3010, Victoria, Australia.

Author's contributions
GA, AK, JZ, and MI conceived and designed the experiments. GA implemented the software and performed the experiments. GA, AK, JZ, and MI analysed the data. All authors read and approved the manuscript.

Received: 9 January 2012 Accepted: 10 May 2012
Published: 10 May 2012

References

1. Manolio TA, Collins FS, Cox NJ, Goldstein DB, Hindorff LA, Hunter DJ, McCarthy MI, Ramos EM, Cardon LR, Chakravarti A, Cho JH, Guttmacher AE, Kong A, Kruglyak L, Mardis E, Rotimi CN, Slatkin M, Valle D, Whittemore AS, Boehnke M, Clark AG, Eichler EE, Gibson G, Haines JL, Mackay TF, McCarroll SA, Visscher PM: **Finding the missing heritability of complex diseases.** *Nature* 2009, **461:**747–753.
2. Tibshirani R: **Regression Shrinkage and Selection via the Lasso.** *J R Statist Soc B* 1996, **58:**267–288.
3. Wu TT, Chen YF, Hastie T, Sobel E, Lange K: **Genome-wide association analysis by lasso penalized logistic regression.** *Bioinformatics* 2009, **25:**714–721.
4. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, Maller J, Sklar P, de Bakker PI, Daly MJ, Sham PC: **PLINK: a tool set for whole-genome association and population-based linkage analyses.** *Am J Hum Genet* 2007, **81:**559–575.
5. Dubois PCA, Trynka G, Franke L, Hunt KA, Romanos J, Curtotti A, Zhernakova A, Heap GAR, Ádány R, Aromaa A, Bardella MT, van den Berg LH, Bockett NA, de la Concha EG, Dema B, Fehrmann RSN, Fernández-Arquero M, Fiatal S, Grandone E, Green PM, Groen HJM, Gwilliam R, Houwen RHJ, Hunt SE, Kaukinen K, Kelleher D, Korponay-Szabo I, Kurppa K, Macmathuna P, Mäki M, Mazzilli MC, Mccann OT, Mearin ML, Mein CA, Mirza MM, Mistry V, Mora B, Morley KI, Mulder CJ, Murray JA, Núñez C, Oosterom E, Ophoff RA, Polanco I, Peltonen L, Platteel M, Rybak A, Salomaa V, Schweizer JJ, Sperandeo MP, Tack GJ, Turner G, Veldink JH, Verbeek WHM, Weersma RK, Wolters VM, Urcelay E, Cukrowska B, Greco L, Neuhausen SL, McManus R, Barisani D, Deloukas P, Barrett JC, Saavalainen P, Wijmenga C, van Heel DA: **Multiple common variants for celiac disease influencing immune gene expression.** *Nat Genet* 2010, **42:**295–304.
6. Friedman J, Hastie T, Tibshirani R: **Regularization Paths for Generalized Linear Models via Coordinate Descent.** *J Stat Soft* 2010, **33:**. [http://www.jstatsoft.org/v33/i01].
7. R Development Core Team: *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing; 2011. [http://www.R-project.org][ISBN3-900051-07-0].
8. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ: **LIBLINEAR: A Library for Large Linear Classification.** *J Mach Learn Res* 2008, **9:**1871–1874.
9. Yu HF, Hsieh CJ, Chang KW, Lin CJ: **Large linear classification when data cannot fit in memory.** In *16th ACM KDD*; 2010.
10. Hoggart CJ, Whittaker JC, Iorio MD, Balding DJ: **Simultaneous analysis of all SNPs in genome-wide and re-sequencing association studies.** *PLoS Genet* 2008, **4:**e1000130.
11. Wray NR, Yang J, Goddard ME, Visscher PM: **The Genetic Interpretation of Area under the ROC Curve in Genomic Profiling.** *PLoS Genet* 2010, **6:**e1000864.
12. Guyon I, Weston J, Barnhill S, Vapnik V: **Gene Selection for Cancer Classification using Support Vector Machines.** *Mach Learn* 2002, **46:**389–422.
13. Hanley JA, McNeil BJ: **The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve.** *Radiology* 1982, **143:**29–36.
14. International HapMap Consortium: **A second generation human haplotype map of over 3.1 million SNPs.** *Nature* 2007, **449:**851–861.
15. International HapMap 3 Consortium: **Integrating common and rare genetic variation in diverse human populations.** *Nature* 2010, **467:**52–58.
16. 1000 Genomes Project Consortium: **A map of human genome variation from population-scale sequencing.** *Nature* 2010, **467:**1061–1073.
17. Friedman J, Hastie T, Höfling H, Tibshirani R: **Pathwise coordinate optimization.** *Ann Appl Statist* 2007, **1:**302–332.
18. Chang KW, Hsieh CJ, Lin CJ: **Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines.** *J Mach Learn Res* 2008, **9:**1369–1398.
19. Van der Kooij AJ: **Prediction Accuracy and Stability of Regression with Optimal Scaling Transformations.** *PhD thesis*. Faculty of Social and Behavioural Sciences, Leiden University; 2007. [http://openaccess.leidenuniv.nl/dspace/handle/1887/12096].
20. Browning SR, Browning BL: **Rapid and accurate haplotype phasing and missing-data inference for whole-genome association studies by use of localized haplotype clustering.** *Am J Hum Genet* 2007, **81:**1084–1097.
21. Howie BN, Donnelly P, Marchini J: **A Flexible and Accurate Genotype Imputation Method for the Next Generation of Genome-Wide Association Studies.** *PLoS Genet* 2009, **5:**e1000529.
22. Li Y, Willer CJ, Ding J, Scheet P, Abecasis GR: **MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes.** *Genet Epidemiol* 2010, **34:**816–834.
23. Price AL, Patterson NJ, Plenge RM, Weinblatt ME, et al.: **Principal components analysis corrects for stratification in genome-wide association studies.** *Nat Genet* 2006, **38:**904–909.

doi:10.1186/1471-2105-13-88
Cite this article as: Abraham *et al.*: SparSNP: Fast and memory-efficient analysis of all SNPs for phenotype prediction. *BMC Bioinformatics* 2012 **13**:88.