problem set no. 3 — due Wednesday 10/21 at 11:59 pm.

problem background. Efficient methods for model fitting, both in a statistical and computational sense, are an imperative for "Big Data". For example, estimation with large-scale models (e.g. thousands to millions of parameters) is (generally) impossible to achieve without the use of *online learning* methods. A very popular method for online learning is Stochastic Gradient Descent (SGD).

This method originates from Sakrison's work [5] that modified the Robbins-Monro procedure [4] for *recursive estimation* of statistical models. For this project we will assume that Y is a univariate random variable and that follows a known distribution model with *unknown* parameters $\boldsymbol{\theta^*} \in \mathbb{R}^p$ and log-likelihood $\ell(\cdots)$, where p is potentially very large. There also exists a dataset $\mathbf{y} = (y_1, y_2, \cdots)$, with covariates $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots\}$, $\mathbf{x}_i \in \mathbb{R}^p$, that is potentially very large. However, we can have *online access* to this dataset i.e., access y_n one-by-one for $n = 1, 2, \cdots$. At every step n we have an estimate $\boldsymbol{\theta}_n$ of $\boldsymbol{\theta^*}$ and upon observing y_n, we update the estimate to $\boldsymbol{\theta}_{n+1}$ and so on. Sakrison [5] considers the following procedure:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + a_n \nabla \ell(\boldsymbol{\theta}_n; y_n, \mathbf{x}_n). \tag{1}$$

It can be shown that Equation (1) estimates $\boldsymbol{\theta}^*$ under some conditions and it has been rediscovered recently as Stochastic Gradient Descent (SGD) in machine learning (ML) [7, 1]. A few important remarks about Equation (1):

- A Newton-Raphson approach would need to calculate $\mathcal{H}_\ell(\boldsymbol{\theta}_n)^{-1} \nabla \ell(\boldsymbol{\theta}_n; \mathbf{y}, \mathbf{X})$, i.e. use inverses of the Hessian and evaluate the derivative at *all datapoints*. This is a *batch training* method and it is generally un-suited for large-scale learning problems. In constrast, SGD is not making expensive matrix inversions (a_n is one-dimensional) and evaluates the log-likelihood at the *current* data point y_n. It is thus an *online learning* method.
- The parameter $a_n \in \mathbb{R}, a_n > 0$ is called a *learning rate* and governs the convergence of the algorithm. According to classical theory (e.g. see [4]) it should be $\sum a_i = \infty$ and $\sum a_i^2 < \infty$. The first condition ensures that we will be able to traverse the parameter space and the second ensures that the estimates will converge and will not oscillate around the limit. One typical way to define the rate is $a_n \propto 1/n$.

There are numerous modifications to the SGD algorithm. One of the most interesting ones is using *implicit updates* that was first applied in the normal linear models [3]. The equation of updates becomes

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n + a_n \nabla \ell(\boldsymbol{\theta}_{n+1}; y_n, \mathbf{x}_n). \tag{2}$$

The key difference is that the update is being calculated in the *future estimate* $\boldsymbol{\theta}_{n+1}$. The estimate appears in both sides of the equation and so the update is called implicit. This is also an online learning method, however performing the update is more costly computationally (sometimes it is impossible). Implicit updates have been shown to be more robust, especially to signal (covariate) noise.

overview. You will implement and run SGD and Implicit algorithms to fit large-scale statistical models, and explore their asympotic bias and variance properties. We will consider Generalized linear models (GLM):

$$\begin{aligned} &\mathbf{x}_n \sim G \\ &\lambda_n = \mathbf{x}_n^\intercal \boldsymbol{\theta}^* \\ &y_n|\lambda_n \sim \exp\left\{\frac{\lambda_n y_n - b(\lambda_n)}{\phi}\right\} \cdot c(y_n, \phi), \end{aligned} \tag{3}$$

where G is some fixed distribution assumed known. In a GLM, the expected value $E\left(y_n|\lambda_n\right)$ is set equal to $E\left(y_n|\lambda_n\right) = h(\lambda_n)$ for some function $h(\cdot)$ that depends on the distributional assumption for Y. The function $h(\cdot)$ is called the *transfer function* and $g(\cdot) = h^{-1}(\cdot)$ is the *link function*, since it links the expected value to predictors, i.e. $g(E\left(y_n|\lambda_n\right)) = \lambda_n = \mathbf{x}_n^\intercal \boldsymbol{\theta}^*$. Also, $\phi > 0$ is the *dispersion parameter* and controls the variance of Y.

question 1. (20 points) Show that the following hold for the GLM of Eqs. (3).

(a) $E\left(y_n|\mathbf{x}_n\right) = h(\mathbf{x}_n^\intercal \boldsymbol{\theta}^*) = b'(\lambda_n)$.

(b) $Var\left(y_n|\lambda_n\right) = \phi \cdot h'(\lambda_n)$.

(c) $\nabla \ell(\boldsymbol{\theta}; y_n, \mathbf{x}_n) = \frac{1}{\phi}\left(y_n - h(\mathbf{x}_n^\intercal \boldsymbol{\theta})\right)\mathbf{x}_n$, derivative w.r.t. $\boldsymbol{\theta}$.

(d) $\mathcal{I}(\boldsymbol{\theta}) = -E\left(\nabla\nabla \ell(\boldsymbol{\theta}; y_n, \mathbf{x}_n)\right) = \frac{1}{\phi} E\left(h'(\mathbf{x}_n^\intercal \boldsymbol{\theta})\mathbf{x}_n \mathbf{x}_n'\right)$.

(e) The function $h(\cdot)$ is nondecreasing.

question 2. (40 points) We will work on the paper by Xu [6] and replicate the experiments in Sections 6.1 and 6.2 that includes `R` implementations of SGD, ASGD and Implicit.

(a) In Section 6.1, work on the toy model set in [6]. Derive the SGD, ASGD (averaged-SGD) and Implicit updates for the model. The averaged-SGD (ASGD) is a variant of SGD that takes averages of recent estimates. It is defined in Algorithm 1 of [6] and it is straightforward to implement. **Replicate** Figure 1 and add the Implicit method as well. You don't need to have the same learning rate for Implicit and standard SGD. Experiment with the learning rate of the Implicit method to make it perform best, and report your findings quantitatively or graphically.

(b) In Section 6.2, work on the toy normal linear model $y_n|\lambda_n \sim \mathcal{N}(\lambda_n, \sigma^2)$, $\lambda_n = \mathbf{x}_n\boldsymbol{\theta}^*$, and $\boldsymbol{\theta}^* \in \mathbb{R}^p$. Xu [6] assumes $p = 100, \sigma^2 = 1, \boldsymbol{\theta}^* = \mathbf{1}$ and $\mathbf{x}_n \sim \mathcal{N}_{100}(0, A)$, where A has a specific eigenvalue form (see paper). Again, derive the SGD, ASGD and Implicit updates. **Replicate** Figure 2 including the Implicit method as well. Note that for standard SGD and the Implicit method, we will assume a rate $a_n = \alpha/n$. Produce the Figure under optimal learning rate parameters α that you will find empirically for two aforementioned methods.

(c) Work on the model of (b) and explore the bias of the estimates of SGD, ASGD and Implicit methods. Implement the following pseudocode:

```
for method in {SGD, Implicit, ASGD}
   for alpha in seq(amin, amax, length.out=10)
      for rep in 1, 2, ..m
         y = sample.data(nsamples=10000)
         theta = run.method(data=y, method=method, to=10000)
         D[method][alpha][rep] =  theta

// plot
for method in {SGD, Implicit, ASGD}
    plot.bias(D[method][.][.])
    plot.variance(D[method][.][.])
```

Make a design (pick m) to use up to **2hr** Odyssey time and at most 50 nodes. The values `amin, amax` are reasonable min-max values that you will pick empirically (e.g., 0.05 to 10), so that your experiment exhibits variability. For the bias, and for each method, you will produce 10 lines (one for each learning rate parameter α) where n is on x-axis and the bias $||\boldsymbol{\theta}_n - \boldsymbol{\theta}^*||$ is on the y-axis, and plot them either on the same plot, or on a grid of plots. Note that to compute the bias $||\boldsymbol{\theta}_n - \boldsymbol{\theta}^*||$ you will average over your m samples.

(d) Use the dataset in the previous part to obtain plots for the variance. In particular, for each method you will produce 10 lines (one for each learning rate parameter α) where n is on x-axis and the trace of the empirical variance matrix $\text{trace}(\text{Var}(\boldsymbol{\theta}_n))$ is on the the y-axis, and plot them either on the same plot, or on a grid of plots. Note that to compute $\text{trace}(\text{Var}(\boldsymbol{\theta}_n))$ you will average over your m samples.

(e) Investigate the relationship of the asympotic variance with the learning rate. In particular, check whether the asymptotic variance matrix of standard and implicit SGD methods with a learning rate $\alpha_n = \alpha/n$ satisfies

$$n \cdot \boldsymbol{\Sigma}_n \to \alpha^2(2\alpha\mathcal{I}(\boldsymbol{\theta}^*) - \mathbf{I})^{-1}\mathcal{I}(\boldsymbol{\theta}^*), \tag{4}$$

where $\mathbf{I}$ is the identity matrix and $\mathcal{I}(\boldsymbol{\theta}^*)$ is the Fisher information matrix at the true parameters, and $\boldsymbol{\Sigma}_n = \text{Var}(\boldsymbol{\theta}_n)$. Also investigate the claim that the iterates from ASGD are asymptotically optimal i.e., have a variance $(1/n)\mathcal{I}(\boldsymbol{\theta}^*)^{-1}$. To compare whether matrices A, B are "close", you might want to check $||A - B||$ or trace$(A - B)$. Derive an optimal learning rate based on the equation above (e.g., the value that minimizes the trace) and compare this value with the optimal value you found empirically in (b).

question 3. (40 points) We will work on the paper by Friedman et. al. [2] and replicate the experiments in Section 5.1.

(a) In that paper the authors describe their package `glmnet` that fits generalized linear models through the "elastic net". Explain what the elastic net is, and what is the role of the quadratic term in the regularized problem. Also explain how `glmnet` is exploiting sparsity to speed up calculations.

(b) In Section 5.1, the authors set up a simulation framework on a linear regression problem of arbitrary dimension. Use their code and the `glmnet` package to reproduce Table 1 in the paper.

(c) Implement the estimation method for the same problem using standard SGD, and create a new Table 1 for this method. Compare with the results from `glmnet` and comment on your findings.

(d) Try to increase the experiment size as much as you can by comparing over larger sample sizes N (e.g. $N = 10^6$) and more covariates e.g., $p = 10^3$.

Note. The `glmnet` package returns a grid of estimates for many different values of regularization parameter λ. In order to compare with the SGD methods, e.g. in terms of MSE, take the median value that is calculated over that grid.

* * *

what to submit. You will submit 1 zip file to stat221.harvard.fall2014@gmail.com. Do not include datasets. The zip file should be named `username_ps3.zip` and will contain `R` + `slurm` code to run the simulations, and the required plots and tables. In summary, we need 1 Figure for 2(a), 1 Figure for2 (b), 1 or more figures for each method for bias 2(c), and 1 or more figures for each method for variance 2(d). For problem (3), you will need to submit 3 tables, in addition to your code. It is recommended to use a naming scheme for your R scripts such as `ps2_username_sgd.R`, and so on.

Happy Coding!

References

[1] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[2] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[3] Jin-Ichi Nagumo and Atsuhiko Noda. A learning method for system identification. *Automatic Control, IEEE Transactions on*, 12(3):282–287, 1967.

[4] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.

[5] David J Sakrison. Efficient recursive estimation; application to estimating the parameters of a covariance function. *International Journal of Engineering Science*, 3(4):461–483, 1965.

[6] Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

[7] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.