Stat 221 Problem Set 3

Albert Young and Marco Gentili

22 October 2014

1 Question 1

1.1 Part a

Show $\mathbb{E}(y_n|\mathbf{x}_n) = b'(\lambda_n)$:

$$\begin{aligned} f(y_n|\lambda_n) &= \exp\left(\frac{\lambda_n y_n - b(\lambda_n)}{\phi}\right) \cdot c(y_n, \phi) \\ \log f(y_n|\lambda_n) &= \frac{\lambda_n y_n - b(\lambda_n)}{\phi} + c(y_n, \phi) \\ \frac{\partial}{\partial \lambda_n} \log f(y_n|\lambda_n) &= \frac{y_n - b'(\lambda_n)}{\phi} \\ y_n &= \phi \frac{\partial}{\partial \lambda_n} \log f(y_n|\lambda_n) + b'(\theta) \end{aligned}$$

Computing the expected value:

$$\begin{aligned} \mathbb{E}(y_n|\mathbf{x_n}) &= \int_{y_n} y_n f(y_n|\lambda_n) \mathrm{d}y_n \\ &= \int_{y_n} \left[\phi \frac{\partial}{\partial \lambda_n} \log f(y_n|\lambda_n) + b'(\lambda_n)\right] f(y_n|\lambda_n) \mathrm{d}y_n \\ &= \phi \int_{y_n} \frac{f(y_n|\lambda_n)}{f(y_n|\lambda_n)} \frac{\partial f(y_n|\lambda_n)}{\partial \lambda_n} \mathrm{d}y_n + \int_{y_n} b'(\lambda_n) f(y_n|\lambda_n) \mathrm{d}y_n \\ &= \phi \int_{y_n} \frac{\partial f(y_n|\lambda_n)}{\partial \lambda_n} \mathrm{d}y_n + \int_{y_n} b'(\lambda_n) f(y_n|\lambda_n) \mathrm{d}y_n \end{aligned}$$

Assuming that we can interchange $\int_{y_n}$ and $\frac{\partial}{\partial \lambda_n}$:

$$\begin{aligned} &= \phi \frac{\partial}{\partial \lambda_n} \int_{y_n} f(y_n|\lambda_n) \mathrm{d}y_n + \int_{y_n} b'(\lambda_n) f(y_n|\lambda_n) \mathrm{d}y_n \\ &= \phi \frac{\partial}{\partial \lambda_n} 1 \mathrm{d}y_n + \int_{y_n} b'(\lambda_n) f(y_n|\lambda_n) \mathrm{d}y_n \\ &= 0 + \int_{y_n} b'(\lambda_n) f(y_n|\lambda_n) \mathrm{d}y_n \\ &= b'(\lambda_n) \int_{y_n} f(y_n|\lambda_n) \mathrm{d}y_n \\ &= b'(\lambda_n) \end{aligned}$$

1.2 Part b

Show $Var(y_n|\lambda_n) = \phi \cdot h'(\lambda_n)$:

$$\begin{aligned} 0 &= \int_{y_n} \frac{\partial^2}{\partial \lambda_n^2} f(y_n|\lambda_n) \mathrm{d}y_n \\ &= \int_{y_n} \left[\left(\frac{\partial^2}{\partial \lambda_n^2} \log f \right) f + \left(\frac{\partial}{\partial \lambda_n} \log f \right)^2 \right] \mathrm{d}y_n \\ &= \mathbb{E}\left(\frac{\partial^2}{\partial \lambda_n^2} \log f \right) + \mathbb{E}\left(\frac{\partial}{\partial \lambda_n} \log f \right)^2 \end{aligned}$$

Thus:

$$\begin{aligned} \mathbb{E}\left(\frac{y_n - b'(\lambda_n)}{\phi} \right)^2 &= \frac{b''(\lambda_n)}{\phi} \\ Var(y_n|\lambda_n) &= \phi \cdot h'(\lambda_n) \end{aligned}$$

1.3 Part c

We have from before that $f(y_n|\lambda_n) = f(y_n|x_n, \theta_n) = \exp\left(\frac{x_n^T \theta y_n - b(x_n^T \theta)}{\phi} \right) \cdot c(y_n, \phi)$.

We know that $\ell(\theta; y_n, x_n) = \log p(x_n, y_n|\theta)$ up to a normalization constant.

$\log f(y_n, x_n|\theta) = \frac{x_n^T \theta y_n - b(x_n^T \theta)}{\phi} + c(y_n, \theta)$. Taking the derivative of this with respect to θ, we get $\nabla \ell(\theta; y_n, x_n) = \frac{y_n - h(x_n^T \theta)}{\phi} x_n$ as desired.

1.4 Part d

$\mathcal{I}$ is the observed information, and is defined to be $-E(\nabla \nabla \ell(\theta; y_n, x_n))$. We know $\nabla \ell(\theta; y_n, x_n)$ from before, so taking the gradient again, we get that

$$\nabla \ell(\theta; y_n, x_n) = \frac{-h'(x_n^T \theta) x_n x_n^T}{\phi}$$

so that

$$\mathcal{I}(\theta) = -E\left(\frac{(-h'(x_n^T \theta) x_n x_n^T}{\phi} \right) = \frac{1}{\phi} E(h'(x_n^T \theta) x_n x_n^T)$$

1.5 Part e

To show that h is nondecreasing, it is sufficient to show that $h' \geq 0$. From part b, we have that $h' = Var(y_n|\lambda_n)/\phi \geq 0$ since the variance of a random variable is always non-negative and $\phi > 0$.

2 Question 2

2.1 Part a

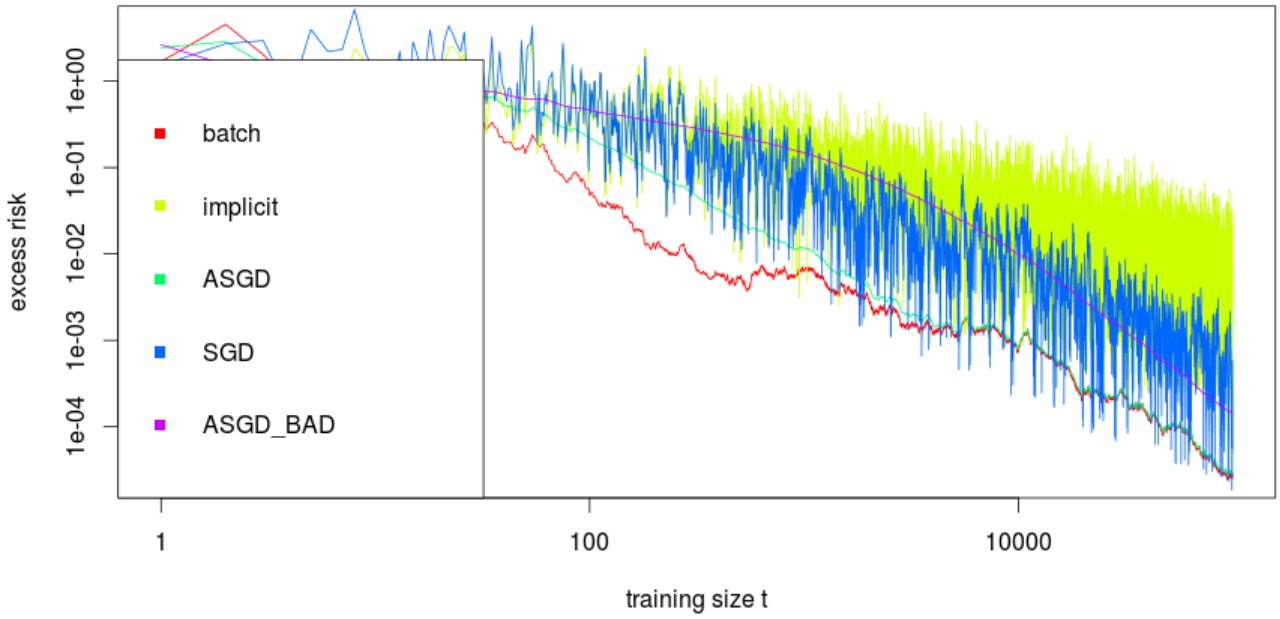
Our loss function is $(\theta - x)^T A (\theta - x)$.

SGD: For stochastic gradient descent, we take the derivative of this with respect to θ to get that $\theta_{n+1} = \theta_n + \alpha_n A(x_n - \theta_n)$.

ASGD: We apply the same updates as SGD, but keep track of the running average of θ.

Implicit: We now have that $\theta_{n+1} = \theta_n + \alpha_n A(x_n - \theta_{n+1})$. Rearranging, we get that $(I + \alpha_n A)\theta_{n+1} = \theta_n + \alpha_n A x_n$, and multiplying both sides by $(I + \alpha_n A)^{-1}$, we get that our updates should be

$$\theta_{n+1} = (I + \alpha_n A)^{-1}(\theta_n + \alpha_n A x_n)$$



We can see that *ASGD_BAD* performs worse than SGD, showing that choosing an appropriate learning rate matters a lot for stochastic gradient descent. Implicit and SGD are quite noisy, which is reasonable given that each new element processed significantly affects the θ value.

2.2 Part b

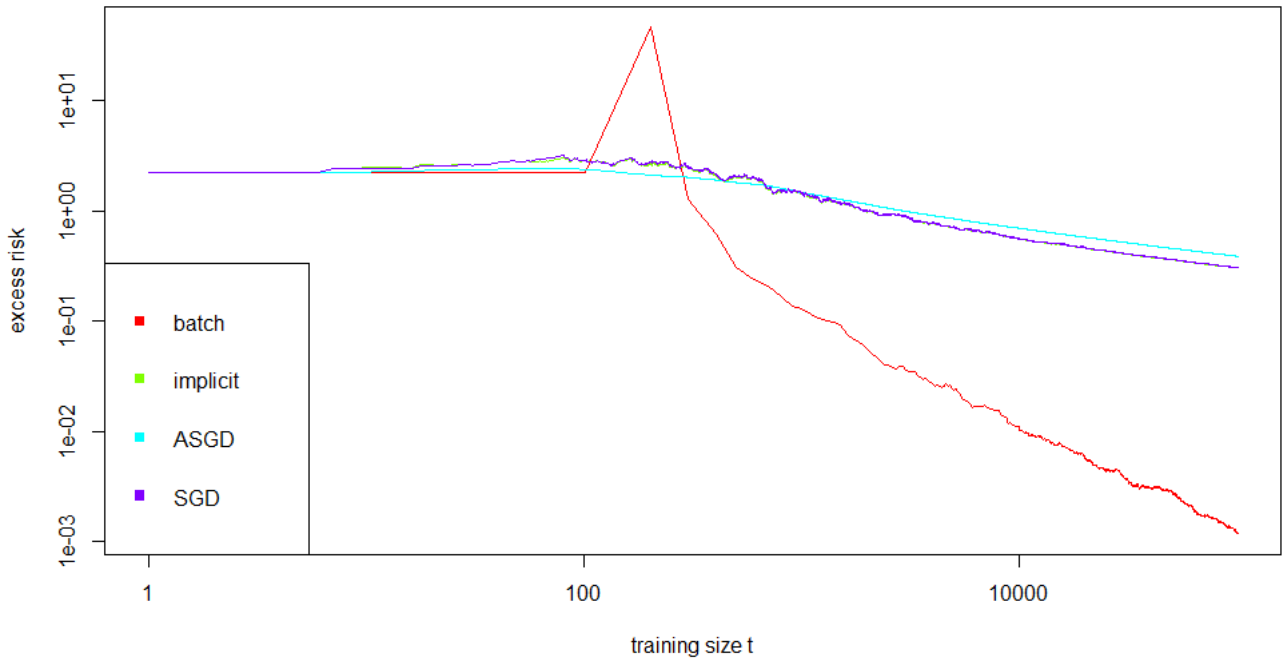
Our loss function is $(y - X\theta)^T(y - X\theta)$.

SGD: for stochastic gradient descent, we take the derivate of this with respect to θ to get that $\theta_{n+1} = \theta_n + \alpha_n X_n^T(y - X_n\theta_n)$.

ASGD: Once again, we apply the same updates as SGD but keep track of the running average of θ.

Implicit: We now have that $\theta_{n+1} = \theta_n + \alpha_n X_n^T(y - X_n\theta_{n+1})$. Rearranging, we get that $(I + \alpha_n X_n^T X_n)\theta_{n+1} = \theta_n + \alpha_n X_n^T y$. Multiplying both sides by $(I + \alpha_n X_n^T X_n)^{-1}$, we get that

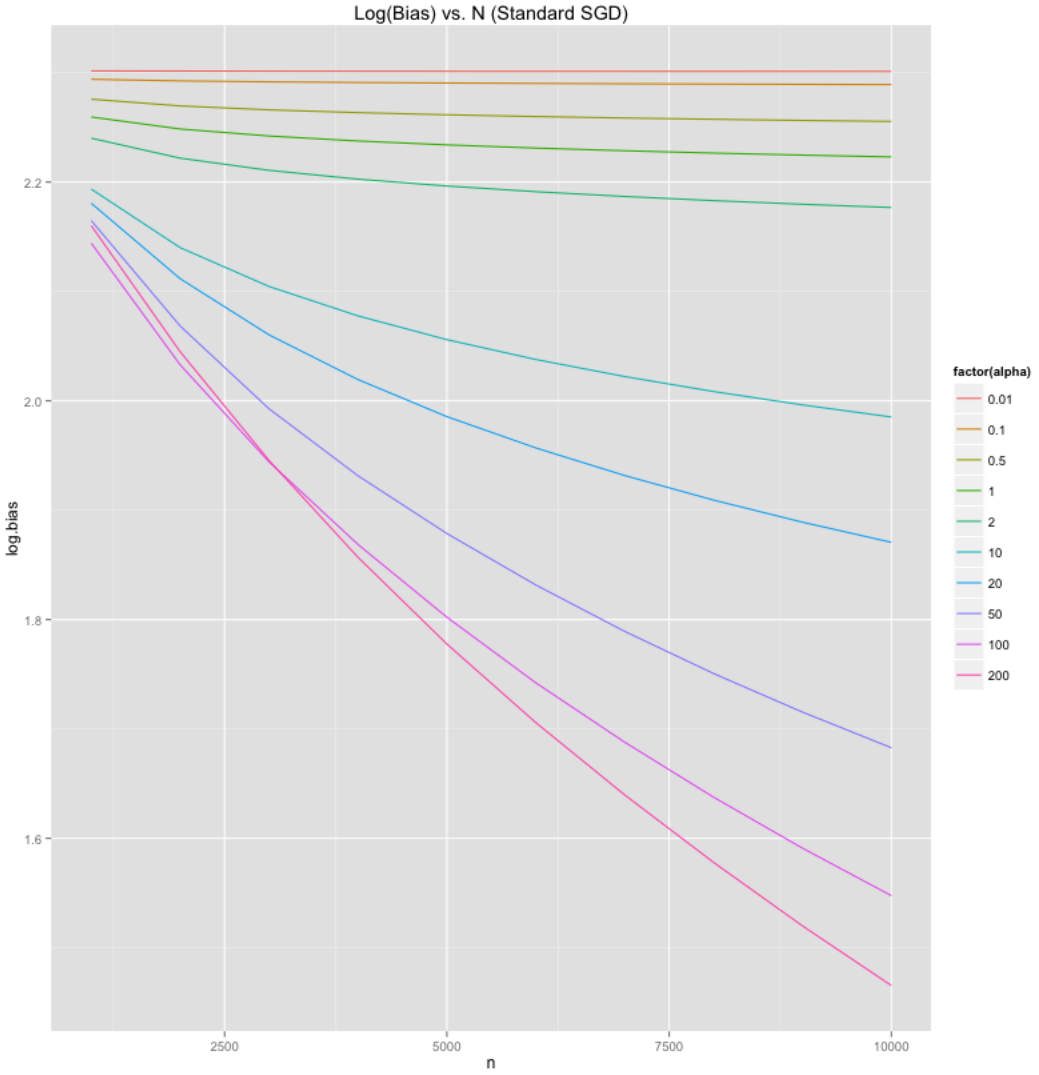
$$\theta_{n+1} = (I + \alpha_n X_n^T X_n)^{-1}(\theta_n + \alpha_n X_n^T y)$$

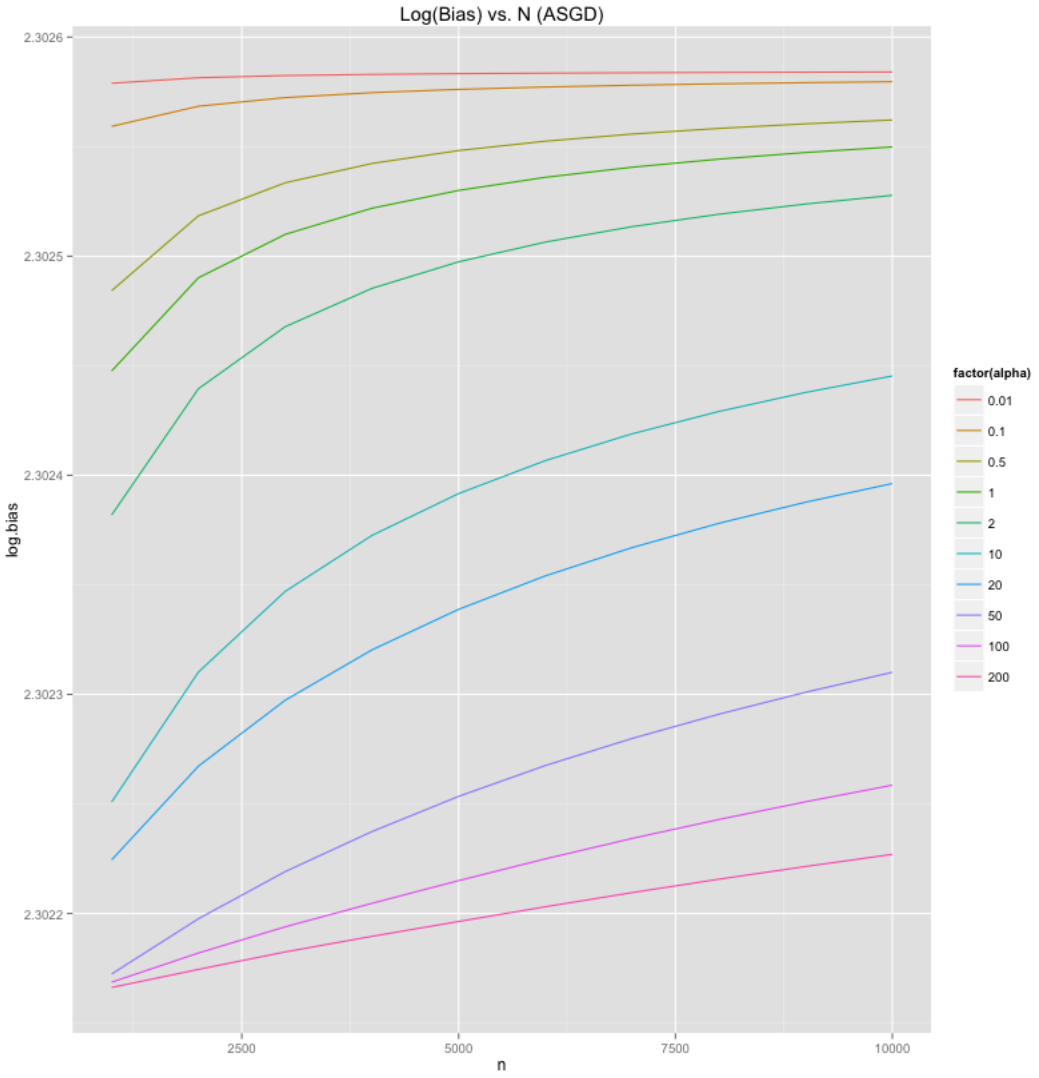


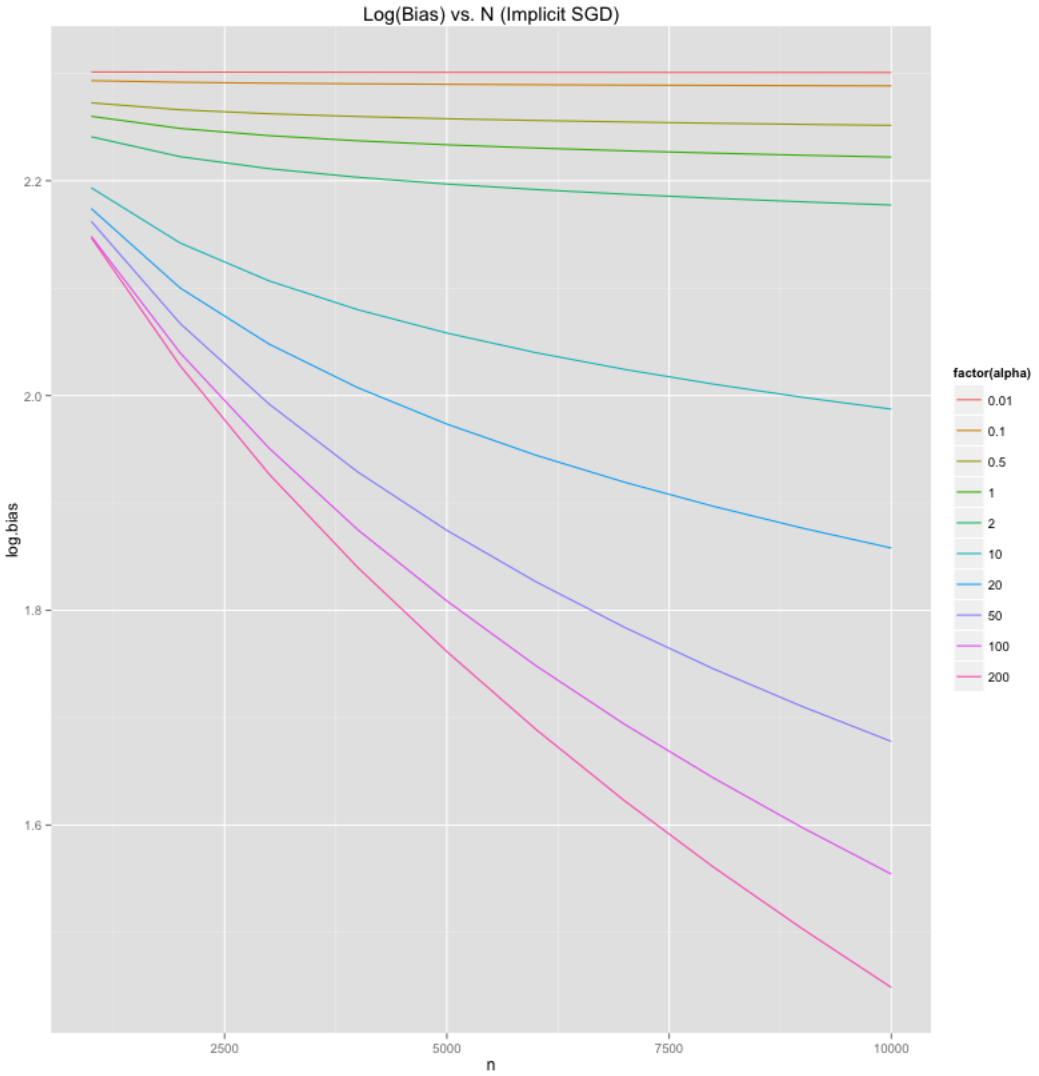
We see a sharp jump in the excess risk for the batch at a size of 100 since the matrix operations aren't valid until $n > p$, which in this case is 100. Since we also compute the batch updates in increments of 100, we don't see an actual value until training size is 200, which is the high point in the excess risk. The other lines decrease quite slowly, which seems to show that the learning rate should be increased for convergence in a reasonable amount of time.

2.3 Part c

For our design, we chose $m = 5$, allowing for each of 50 nodes to calculate 1 iteration with 1 *alpha*.

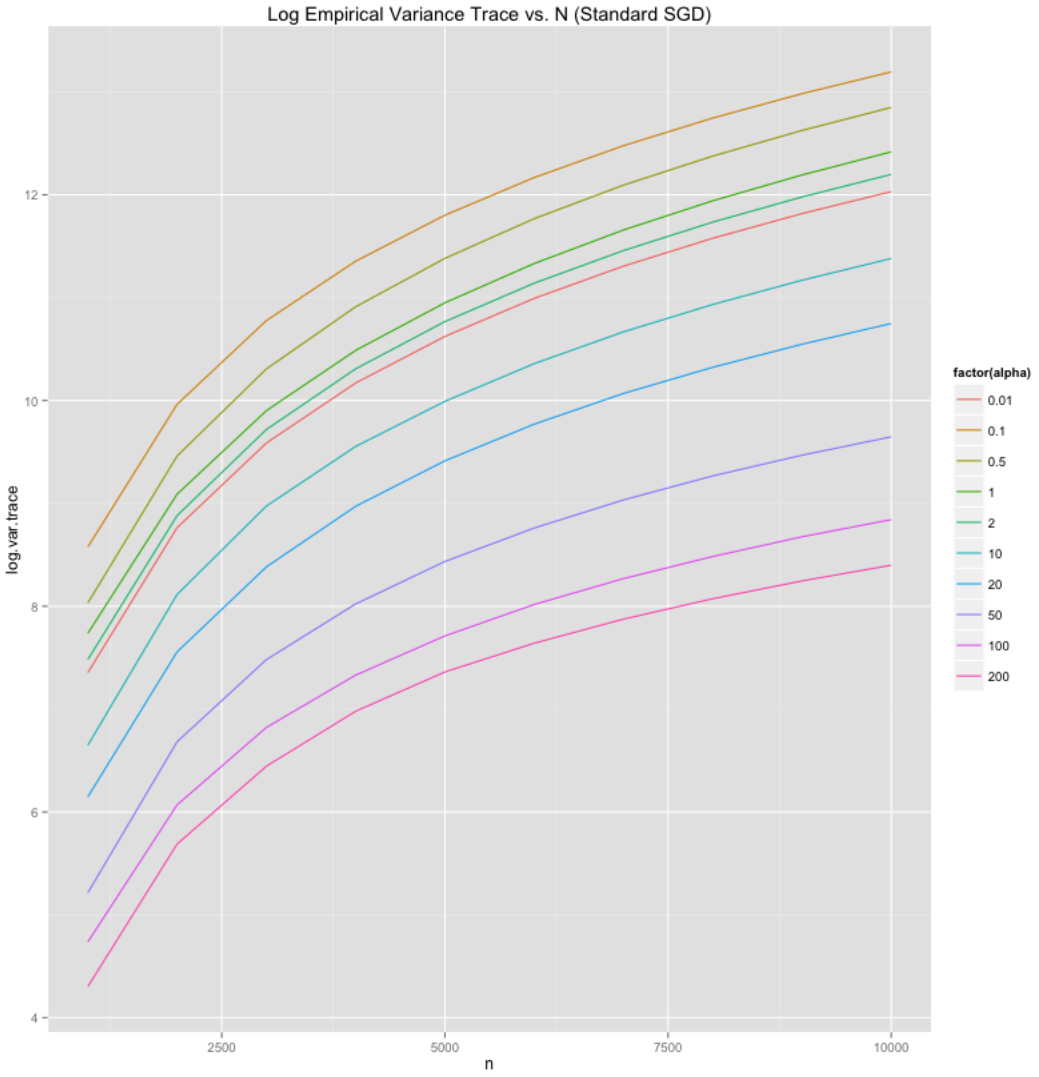


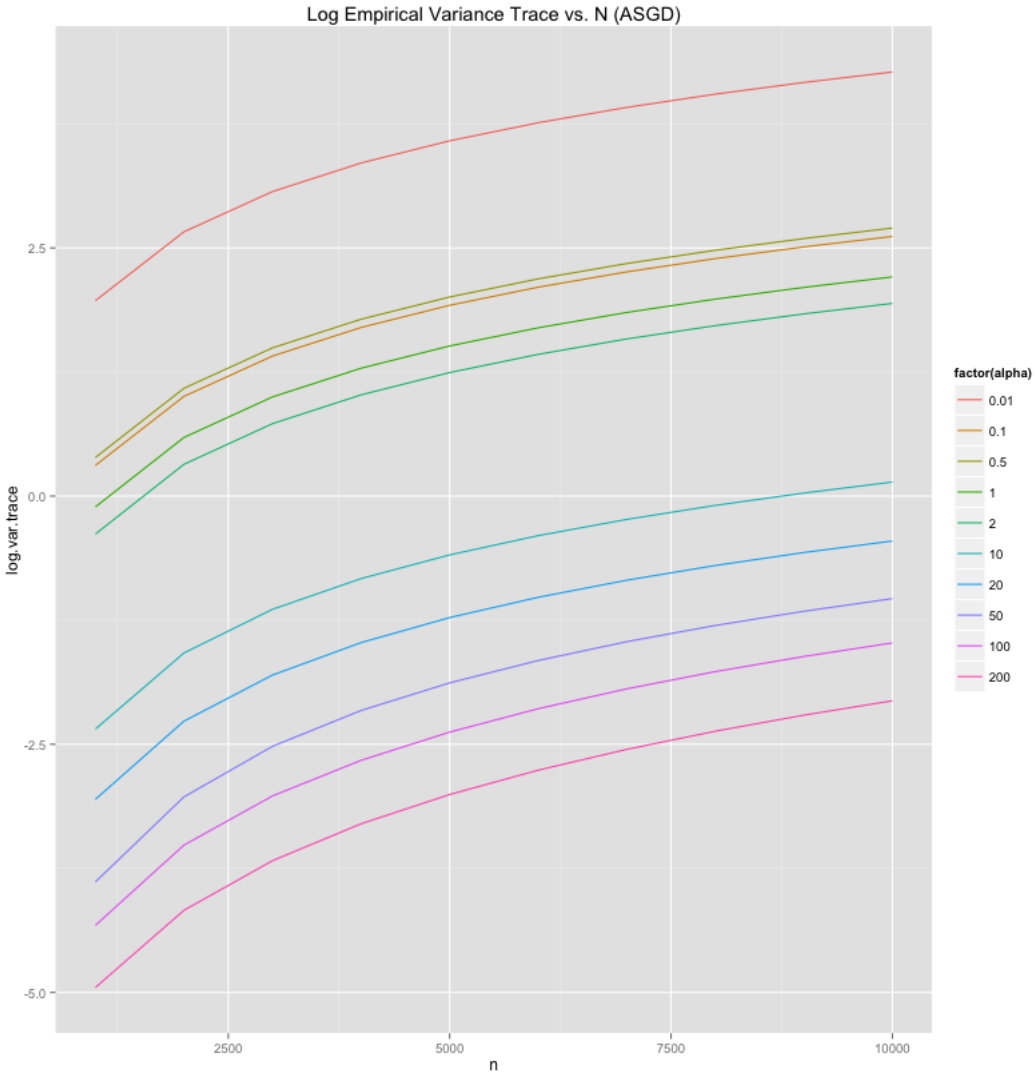

Log(Bias) vs. N (ASGD)
log.bias
2.3026
2.3025
2.3024
2.3023
2.3022
2500
5000
7500
10000
n
factor(alpha)
0.01
0.1
0.5
1
2
10
20
50
100
200

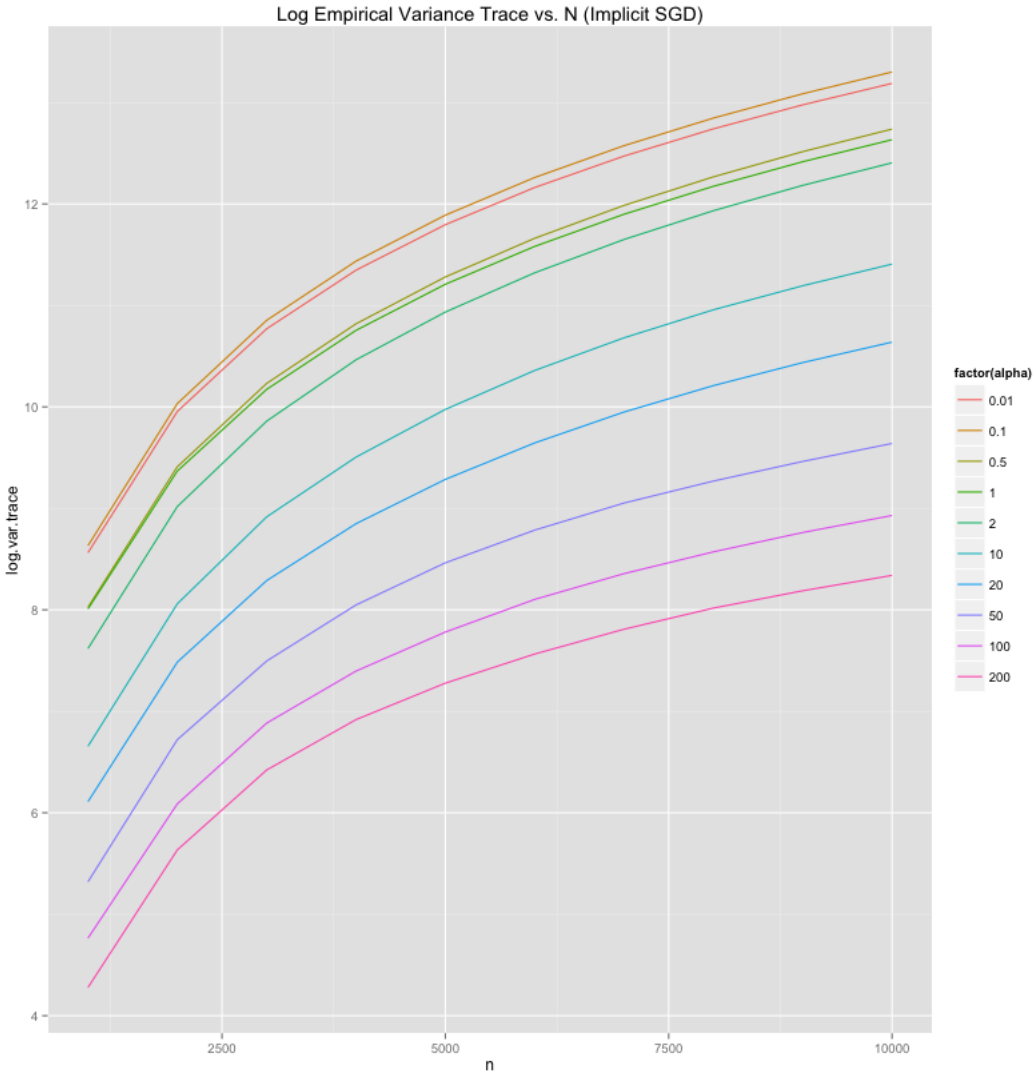


For higher α the bias is lower and decreases more steeply. For some reason, ASGD exhibits increasing bias for larger values of n.

2.4 Part d


Log Empirical Variance Trace vs. N (Standard SGD)
log.var.trace
n
12
10
8
6
4
2500
5000
7500
10000
factor(alpha)
0.01
0.1
0.5
1
2
10
20
50
100
200

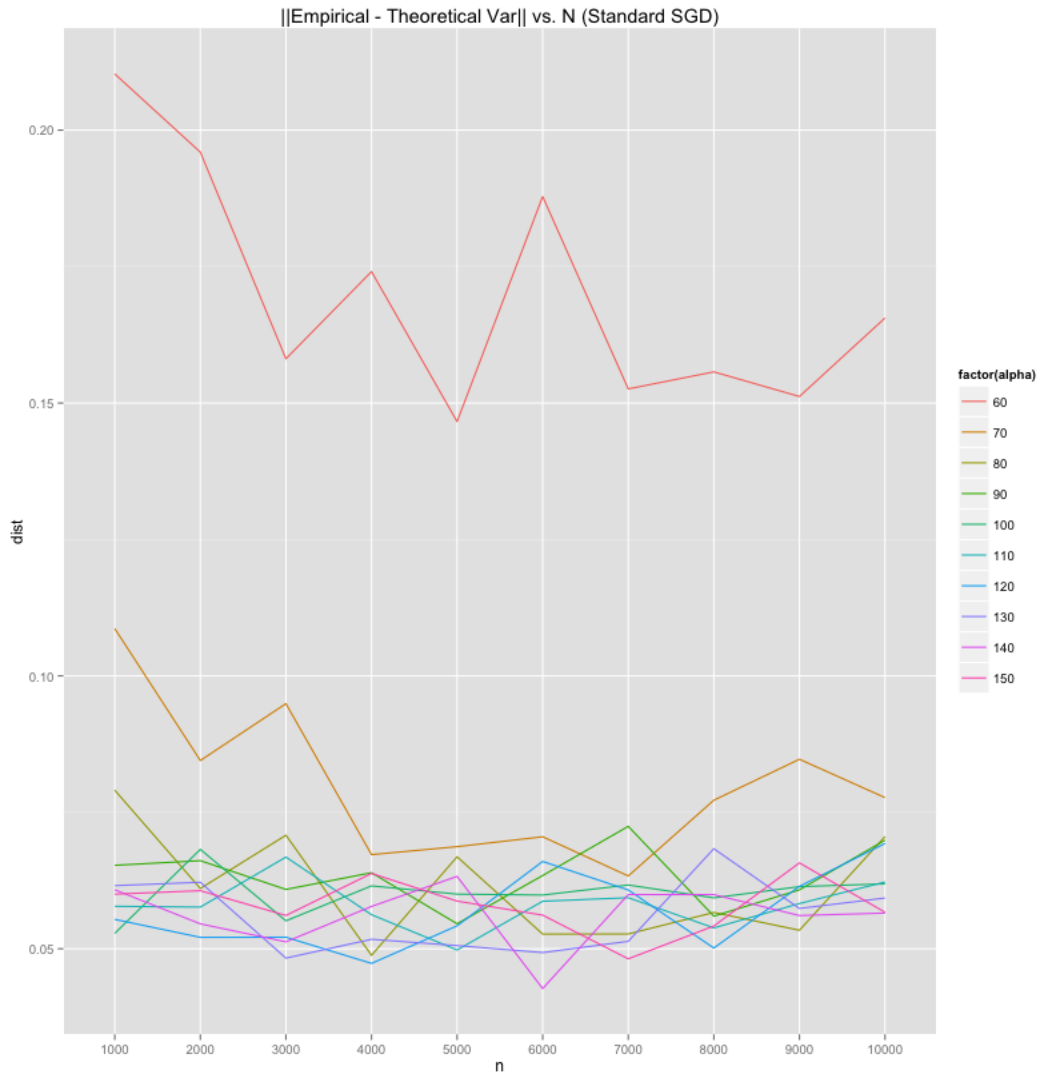

Log Empirical Variance Trace vs. N (ASGD)
log.var.trace
2.5
0.0
-2.5
-5.0
2500
5000
7500
10000
n
factor(alpha)
0.01
0.1
0.5
1
2
10
20
50
100
200



For higher α the empirical variance is smaller.

2.5 Part e

In this part we chose to perform SGD for 100 replicates for each combination of α and n.



It appears that asymptotically, the variance converges to the theoretical value, whihc faster convergence observed for larger values of α.

3 Question 3

3.1 Part a

The elastic net is a regularized regression method for correlated variables, which uses a penalty that is part ℓ_1, part ℓ_2. The role of the quadratic term is to shrink the coefficients of correlated predictors towards each other, as in ridge regression. glmnet exploits sparsity by storing sparse feature matrices in a space-efficient sparse column format and by using coordinate descent to sum over only the non-zero entries in computing inner-product operations in either the naive or covariance updates.

3.2 Part b and Part c

The numbers in the below tables indicate timings of the different methods. glmnet was run using default parameters and the timing is averaged over three runs. $N = 1000, p = 100$

	rho					
	0	0.1	0.2	0.5	0.9	0.95
glmnet (cov)	0.015000	0.015000	0.01567	0.01567	0.02233	0.02833
glmnet (naive)	0.047667	0.047000	0.05033	0.06800	0.18500	0.27500
sgd (implicit)	0.016333	0.016000	0.01700	0.01567	0.01633	0.01800
sgd (standard)	0.009667	0.009667	0.01000	0.00900	0.00900	0.01000

$N = 5000, p = 100$

	rho					
	0	0.1	0.2	0.5	0.9	0.95
glmnet (cov)	0.06367	0.052	0.03200	0.04467	0.0460	0.05667
glmnet (naive)	0.25033	0.214	0.13500	0.24700	0.7207	1.11600
sgd (implicit)	0.10467	0.101	0.05833	0.08167	0.0800	0.09400
sgd (standard)	0.06500	0.054	0.03200	0.04500	0.0440	0.05067

$N = 100, p = 1000$

	rho					
	0	0.1	0.2	0.5	0.9	0.95
glmnet (cov)	0.043000	0.072667	0.083333	0.092000	0.318667	0.292333
glmnet (naive)	0.026000	0.036000	0.043667	0.060000	0.123000	0.164667
sgd (implicit)	0.003333	0.004000	0.005333	0.005000	0.009667	0.008667
sgd (standard)	0.002000	0.003333	0.003667	0.002333	0.004667	0.005667

$N = 100, p = 5000$

	rho					
	0	0.1	0.2	0.5	0.9	0.95
glmnet (cov)	0.57867	0.52600	0.65100	0.75733	1.11467	1.12967
glmnet (naive)	0.26467	0.26467	0.35800	0.34833	0.29767	0.57833
sgd (implicit)	0.03367	0.03400	0.03067	0.03167	0.02867	0.02600
sgd (standard)	0.01733	0.01667	0.02600	0.01667	0.01567	0.01567

$N = 100, p = 20000$

	rho					
	0	0.1	0.2	0.5	0.9	0.95
glmnet (cov)	1.99733	2.5900	1.07567	3.2700	5.8863	6.7387
glmnet (naive)	0.75333	0.7240	0.39333	0.9343	1.0023	1.6597
sgd (implicit)	0.14233	0.1647	0.05700	0.1170	0.1217	0.1113
sgd (standard)	0.05767	0.0800	0.03333	0.0680	0.0580	0.0660

$N = 100, p = 50000$

	rho					
	0	0.1	0.2	0.5	0.9	0.95
glmnet (cov)	3.5303	3.3697	3.255	7.5460	6.8167	8.3593
glmnet (naive)	1.1217	1.2350	1.259	2.4400	1.2237	1.8030
sgd (implicit)	0.1837	0.1737	0.160	0.4557	0.1957	0.1897
sgd (standard)	0.1343	0.1023	0.094	0.3037	0.1387	0.1260

The results are similar to those observed in Friedman et al. SGD is generally faster than glmnet, especially for lower values of n. glmnet slows significantly under high correlation, while SGD is little affected. The naive glmnet procedure performs better than the covariance update for $p \geq 1000$.

3.3 Part d

$N = 1000, p = 50000$

			method		
rho		glmnet (cov)	glmnet (naive)	sgd (implicit)	sgd (standard)
0	time	222.50233	28.55233	2.75733	2.13833
	mse	0.01401	0.01405	0.01839	0.02152
0.1	time	272.63667	31.26933	3.09433	2.40700
	mse	0.01408	0.01411	0.01896	0.02120
0.2	time	250.12100	32.12333	3.09200	2.32700
	mse	0.01398	0.01407	0.01941	0.02102
0.5	time	475.49767	61.06633	4.83933	3.40600
	mse	0.01413	0.01424	0.02190	0.02487
0.9	time	315.00267	80.63867	3.95733	3.01800
	mse	0.01582	0.01588	0.02957	0.09456
0.95	time	245.48833	78.50400	2.79900	2.13300
	mse	0.01775	0.01775	0.03121	0.05827

$N = 10^4, p = 5000$

			method		
rho		glmnet (cov)	glmnet (naive)	sgd (implicit)	sgd (standard)
0	time	222.50233	28.55233	2.75733	2.13833
	mse	0.01401	0.01405	0.01839	0.02152
0.1	time	272.63667	31.26933	3.09433	2.40700
	mse	0.01408	0.01411	0.01896	0.02120
0.2	time	250.12100	32.12333	3.09200	2.32700
	mse	0.01398	0.01407	0.01941	0.02102
0.5	time	475.49767	61.06633	4.83933	3.40600
	mse	0.01413	0.01424	0.02190	0.02487
0.9	time	315.00267	80.63867	3.95733	3.01800
	mse	0.01582	0.01588	0.02957	0.09456
0.95	time	245.48833	78.50400	2.79900	2.13300
	mse	0.01775	0.01775	0.03121	0.05827

$N = 10^5, p = 1000$

			method		
rho		glmnet (cov)	glmnet (naive)	sgd (implicit)	sgd (standard)
0	time	$1.836e+02$	$4.587e+01$	11.028667	$7.626e+00$
	mse	$4.884e-03$	$4.884e-03$	0.019136	$2.476e-02$
0.1	time	$1.487e+02$	$3.859e+01$	8.153000	$5.766e+00$
	mse	$4.988e-03$	$4.988e-03$	0.018638	$2.437e-02$
0.2	time	$9.391e+01$	$2.503e+01$	6.501000	$4.719e+00$
	mse	$5.099e-03$	$5.099e-03$	0.018223	$2.455e-02$
0.5	time	$2.058e+02$	$6.696e+01$	15.671000	$1.137e+01$
	mse	$5.650e-03$	$5.650e-03$	0.016126	$2.352e-02$
0.9	time	$4.020e+02$	$3.318e+02$	21.623000	$1.555e+01$
	mse	$1.055e-02$	$1.055e-02$	0.010139	$3.860e+04$
0.95	time	$1.239e+02$	$1.558e+02$	9.325000	$6.719e+00$
	mse	$1.811e-02$	$1.811e-02$	0.008996	$1.986e+01$

$N = 10^6, p = 100$

		method			
rho		glmnet (cov)	glmnet (naive)	sgd (implicit)	sgd (standard)
0	time	$1.836e+02$	$4.587e+01$	11.028667	$7.626e+00$
	mse	$4.884e-03$	$4.884e-03$	0.019136	$2.476e-02$
0.1	time	$1.487e+02$	$3.859e+01$	8.153000	$5.766e+00$
	mse	$4.988e-03$	$4.988e-03$	0.018638	$2.437e-02$
0.2	time	$9.391e+01$	$2.503e+01$	6.501000	$4.719e+00$
	mse	$5.099e-03$	$5.099e-03$	0.018223	$2.455e-02$
0.5	time	$2.058e+02$	$6.696e+01$	15.671000	$1.137e+01$
	mse	$5.650e-03$	$5.650e-03$	0.016126	$2.352e-02$
0.9	time	$4.020e+02$	$3.318e+02$	21.623000	$1.555e+01$
	mse	$1.055e-02$	$1.055e-02$	0.010139	$3.860e+04$
0.95	time	$1.239e+02$	$1.558e+02$	9.325000	$6.719e+00$
	mse	$1.811e-02$	$1.811e-02$	0.008996	$1.986e+01$

For larger values of N, glmnet takes significantly longer than SGD, but produces more accurate estimates (smaller MSE). Implicit SGD takes slightly longer to run than the standard SGD but produces better estimates, especially for larger ρ.