Statistics 221 Final Project: C++ team

Adam Petcher, Brandon Sim, Andrew Liu, Jason Sclar, Rohit Ramani

December 17, 2013

1 Compare MLE to SGD

MLE methods such Newton-Raphson compute the gradient of the log-likelihood, $\nabla\ell$ for all observations in one step. While this works well for small samples, it is too computationally expensive to be done with especially large amounts of data. Sakrison's method relies on the assumption that the expectation of the gradient for a single observation is proportional to the gradient for all observations. By the law of large numbers, repeatedly updating using the gradient for single observations will converge to expectations. This makes it possible to do the update step for a single observation at a time and compute the MLE.

The implicit updates use the update step, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + a_t\nabla\ell(\boldsymbol{\theta}_{t+1}; y_t, \boldsymbol{x}_t)$ instead of $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + a_t\nabla\ell(\boldsymbol{\theta}_t; y_t, \boldsymbol{x}_t)$. However the expectation of the gradient is the same for all observations. Therefore $\nabla\ell(\boldsymbol{\theta}_{t+1}; y_t, \boldsymbol{x}_t) = \nabla\ell(\boldsymbol{\theta}_t; y_t, \boldsymbol{x}_t)$. Since the implicit method has the same expectation as Sakrison's method, it must also compute the MLE.

2 GLM Proofs

(a) Show $E(y_t|\boldsymbol{x}_t) = h(\boldsymbol{x}_t^T\boldsymbol{\theta}^*) = b'(\eta_t)$

We start with the moment generating function, which we solve for using LOTUS

$$\begin{aligned}
M_Y(t) &= E[e^{tY}] \\
&= \int_Y \exp(ty_k) f(y_k|\eta_k) dy_k \\
&= \int_Y \exp(ty_k) \exp\left(\frac{\eta_t y_t - b(\eta_t)}{\phi}\right) \cdot c(y_t, \phi) dy_k \\
&= \int_Y c(y_t, \phi) \exp\left(ty_k + \frac{\eta_t y_t}{\phi} - \frac{b(\eta_t)}{\phi}\right) dy_k \\
&= \int_Y c(y_t, \phi) \exp\left(\left(\frac{\eta_t y_t + \phi t}{\phi}\right) y_k - \frac{b(\eta_t)}{\phi}\right) dy_k \\
&= \int_Y c(y_t, \phi) \exp\left(\left(\frac{\eta_t y_t + \phi t}{\phi}\right) y_k - \frac{b(\eta_k + \phi t)}{\phi} + \frac{b(\eta_k + \phi t)}{\phi} - \frac{b(\eta_t)}{\phi}\right) dy_k \\
&= \int_Y \exp\left(\frac{b(\eta_k + \phi t) - b(\eta_t)}{\phi}\right) c(y_t, \phi) \exp\left(\frac{(\eta_t y_t + \phi t) y_k - b(\eta_k + \phi t)}{\phi}\right) dy_k \\
&= \exp\left(\frac{b(\eta_k + \phi t) - b(\eta_t)}{\phi}\right) \int_Y c(y_t, \phi) \exp\left(\frac{(\eta_t y_t + \phi t) y_k - b(\eta_k + \phi t)}{\phi}\right) dy_k
\end{aligned}$$

The remaining integral is the conditional pdf of $y|\eta_k + \phi t$ and integrates to 1, which leaves the remaining MGF for Y

$$M_Y(t) = \exp\left(\frac{b(\eta_k + \phi t) - b(\eta_t)}{\phi}\right)$$

We then use the MGF to find $E(y_t|\boldsymbol{x}_t)$. We start by finding $M'_Y(t)$

$$\begin{aligned}
M'_Y(t) &= \frac{\partial}{\partial t}\left[\exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)\right] \\
&= \exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)\cdot\frac{b'(\eta_k+\phi t)\phi}{\phi} \\
&= \exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)\cdot b'(\eta_k+\phi t)
\end{aligned}$$

We then evaluate at $t=0$ which is the first moment.

$$\begin{aligned}
E(y_t|\boldsymbol{x}_t) &= M'_Y(0) \\
&= \exp\left(\frac{b(\eta_k+\phi\cdot 0)-b(\eta_t)}{\phi}\right)\cdot b'(\eta_k+\phi\cdot 0) \\
&= \exp\left(\frac{0}{\phi}\right)\cdot b'(\eta_k) \\
&= b'(\eta_k)
\end{aligned}$$

(b) Show $\text{Var}(y_t|\eta_t) = \phi\cdot h'(\eta_t)$
We find the variance using the moment generating function from part (a).

$$\text{Var}(y_k|\eta_k) = E(y_k^2|\eta_k) - [E(y_k|\eta_k)]^2$$

We can use the value of $E(y_k|\eta_k)$ from part (a). We then find the second moment $E(y_k^2|\eta_k)$ with our MGF.

$$\begin{aligned}
M''_Y(t) =& \frac{\partial}{\partial t}\left[\exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)\cdot b'(\eta_k+\phi t)\right] \\
=& \exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)\cdot b'(\eta_k+\phi t)\cdot b'(\eta_k+\phi t)+ \\
& b''(\eta_k+\phi t)\cdot\phi\cdot\exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right) \\
=& \exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)\cdot\left[b'(\eta_k+\phi t)\right]^2 + b''(\eta_k+\phi t)\cdot\phi\cdot\exp\left(\frac{b(\eta_k+\phi t)-b(\eta_t)}{\phi}\right)
\end{aligned}$$

Evaluating at 0 gives

$$\begin{aligned}
E(y_k^2|\eta_k) =& M''_Y(0) \\
=& \exp\left(\frac{b(\eta_k+\phi*0)-b(\eta_t)}{\phi}\right)\left[b'(\eta_k+\phi*0)\right]^2 + b''(\eta_k+\phi*0)\phi\exp\left(\frac{b(\eta_k+\phi*0)-b(\eta_t)}{\phi}\right) \\
=& \exp(\frac{0}{\phi})\left[b'(\eta_k)\right]^2 + \phi\cdot b''(\eta_k)\exp(\frac{0}{\phi}) \\
=& \left[b'(\eta_k)\right]^2 + \phi\cdot b''(\eta_k)
\end{aligned}$$

Putting the two parts together gives the solution

$$\begin{aligned}\text{Var}(y_t|\eta_t) &= E(y_k^2|\eta_k) - [E(y_k|\eta_k)]^2 \\ &= [b'(\eta_k)]^2 + \phi \cdot b''(\eta_k) - [b'(\eta_k)]^2 \\ &= \phi \cdot b''(\eta_k)\end{aligned}$$

From part (a) we know that $b'(\eta_t) = h(\eta)$ therefore $b''(\eta_t) = h'(\eta_t)$ which gives

$$\text{Var}(y_t|\eta_t) = \phi \cdot h'(\eta_t)$$

(c) Show $\nabla\ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t) = \frac{1}{\phi}(y_t - h(\boldsymbol{x}_t^T\theta))\boldsymbol{x}_t$
To find $\nabla\ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t)$, we take the partial derivative with respect to $\boldsymbol{\theta}$. The likelihood is proportional to the pdf.

$$\begin{aligned}L(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t) &\propto \exp\left(\frac{\eta_t y_t - b(\eta_t)}{\phi}\right) \cdot c(y_t, \phi) \\ \ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t) &= \frac{\eta_t y_t - b(\eta_t)}{\phi} \\ \nabla\ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t) &= \frac{\partial}{\partial\boldsymbol{\theta}}\left[\ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t)\right] \\ &= \frac{\partial}{\partial\boldsymbol{\theta}}\left[\frac{\eta_t y_t - b(\eta_t)}{\phi}\right] \\ &= \frac{\partial}{\partial\boldsymbol{\theta}}\left[\frac{\boldsymbol{x}_t^T\boldsymbol{\theta} y_t - b(\boldsymbol{x}_t^T\boldsymbol{\theta})}{\phi}\right] \\ &= \frac{1}{\phi}(\boldsymbol{x}_t y_t - \boldsymbol{x}_t b'(\boldsymbol{x}_t^T\boldsymbol{\theta})) \\ &= \frac{1}{\phi}(y_t - h(\boldsymbol{x}_t^T\boldsymbol{\theta}))\boldsymbol{x}_t\end{aligned}$$

(d) Show $\mathcal{J}(\boldsymbol{\theta}) = -E(\nabla\nabla\ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t)) = \frac{1}{\phi}E(h'(\boldsymbol{x}_t^T\boldsymbol{\theta})\boldsymbol{x}_t\boldsymbol{x}_t')$
To find the Fisher information, we take the negative expectation of the second partial derivative of the log-likelihood with respect to $\boldsymbol{\theta}$. We start with the the solution from part (c).

$$\begin{aligned}\mathcal{J}(\boldsymbol{\theta}) &= -E\Big(\nabla\nabla\ell(\boldsymbol{\theta}; y_t, \boldsymbol{x}_t)\Big) \\ &= -E\Big(\frac{\partial}{\partial\boldsymbol{\theta}}\Big[\frac{1}{\phi}(y_t - h(\boldsymbol{x}_t^T\boldsymbol{\theta}))\boldsymbol{x}_t\Big]\Big) \\ &= -E\Big(-\frac{1}{\phi}\cdot h'(\boldsymbol{x}_t^T\boldsymbol{\theta})\boldsymbol{x}_t\boldsymbol{x}_t^T\Big) \\ &= \frac{1}{\phi}\Big(h'(\boldsymbol{x}_t^T\boldsymbol{\theta})\boldsymbol{x}_t\boldsymbol{x}_t^T\Big)\end{aligned}$$

3 Implementation and Results

(a) **Log-loss.** The log-loss function is given by:

$$L(y, \hat{y}) = \log(1 + \exp(-y\hat{y})) \tag{1}$$

The SGD update is given by:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha_t \nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) \tag{2}$$

where

$$Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) = \log\left(1 + \exp(-y_t \cdot \boldsymbol{x_t^T \theta_t})\right) + \frac{\lambda}{2}\|\boldsymbol{\theta_t}\|^2 \tag{3}$$

The gradient $\nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t})$ can be calculated as follows:

$$\begin{aligned}
\nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) &= \left(\frac{1}{1 + \exp(-y_t \cdot \boldsymbol{x_t^T \theta_t})}\right)\left(\exp(-y_t \cdot \boldsymbol{x_t^T \theta_t})\right)(-y_t \cdot \boldsymbol{x_t}) + \lambda \boldsymbol{\theta_t} \\
&= \frac{-y_t \exp(-y_t \cdot \boldsymbol{x_t^T \theta_t})}{1 + \exp(-y_t \cdot \boldsymbol{x_t^T \theta_t})} \cdot \boldsymbol{x_t} + \lambda \boldsymbol{\theta_t} \\
&= \frac{-y_t}{\exp(y_t \cdot \boldsymbol{x_t^T \theta_t}) + 1} \cdot \boldsymbol{x_t} + \lambda \boldsymbol{\theta_t}
\end{aligned}$$

So, the SGD update for the log-loss function is:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha_t \left(\frac{-y_t}{\exp(y_t \cdot \boldsymbol{x_t^T \theta_t}) + 1} \cdot \boldsymbol{x_t} + \lambda \boldsymbol{\theta_t}\right) \tag{4}$$

The implicit update can be derived in much the same way as above. We have the following:

$$\begin{aligned}
\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} - \alpha_t \nabla Q(\boldsymbol{\theta_{t+1}}, y_t, \boldsymbol{x_t}) \\
&= \boldsymbol{\theta_t} - \alpha_t \left(\frac{-y_t}{\exp(y_t \cdot \boldsymbol{x_t^T \theta_{t+1}}) + 1} \cdot \boldsymbol{x_t} + \lambda \boldsymbol{\theta_{t+1}}\right)
\end{aligned}$$

Note that we cannot find an analytic solution to the above. Instead, we proceed in a different direction. First, for the implicit update, we drop the regularization term for simplicity, as we believe the implicit method would work well even without regularization. Next, we let $h(x) = e^x/(1 + e^x)$ and transform $y_t \in \{-1, 1\}$ to $y_t \in \{0, 1\}$ as follows:

For $y_t = -1$, we have:

$$\begin{aligned}
\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} - \frac{\alpha_t \boldsymbol{x_t}}{1 + \exp\left(-\boldsymbol{x_t^T \theta_{t+1}}\right)} \\
&= \boldsymbol{\theta_t} + \alpha_t \left(y_t - h(\boldsymbol{x_t^T \theta_{t+1}})\right) \boldsymbol{x_t}
\end{aligned}$$

where $y_t = 0$.

For $y_t = 1$, we have:

$$\begin{aligned}
\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + \frac{\alpha_t \boldsymbol{x_t}}{1 + \exp\left(\boldsymbol{x_t^T \theta_{t+1}}\right)} \\
&= \boldsymbol{\theta_t} + \alpha_t \left(y_t - h(\boldsymbol{x_t^T \theta_{t+1}})\right) \boldsymbol{x_t}
\end{aligned}$$

where $y_t = 1$.

To solve $\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \alpha_t \left(y_t - h(\boldsymbol{x_t^T \theta_{t+1}})\right) \boldsymbol{x_t}$, $y_t \in \{0, 1\}$, we can rewrite as:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \xi_t \boldsymbol{x_t} \tag{5}$$

where

$$\xi_t = \alpha_t \left(y_t - h(\boldsymbol{x}_t^T \boldsymbol{\theta_t} + \xi_t \|\boldsymbol{x_t}\|^2) \right) \tag{6}$$

We now can implement any root-finding procedure efficiently, as long as we have some bounds on where the root will be. These bounds we can find easily, as follows: let $r_t = \alpha_t \left(y_t - h(\boldsymbol{x}_t^T \boldsymbol{\theta_t}) \right)$. Then, if $r_t < 0$, then $\xi_t \in [r_t, 0]$; if $r_t > 0$, then $\xi_t \in [0, r_t]$.

Hinge-loss. The hinge loss function is given by:

$$L(y, \hat{y}) = \max(0, 1 - y\hat{y}) \tag{7}$$

The SGD update is given by:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha_t \nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) \tag{8}$$

where

$$Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) = \max(0, 1 - y_t \cdot \boldsymbol{x_t^T \theta_t}) + \frac{\lambda}{2} \|\boldsymbol{\theta_t}\|^2 \tag{9}$$

The gradient $\nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t})$ can be calculated as follows, where we consider two cases depending on the sign of $1 - y_t \cdot \boldsymbol{x_t^T \theta_t}$:

1. If $1 - y_t \cdot \boldsymbol{x_t^T \theta_t} < 0$, then $Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) = \frac{\lambda}{2} \|\boldsymbol{\theta_t}\|^2$. Then,

$$\begin{aligned} \nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) &= \nabla \left(\frac{\lambda}{2} \|\boldsymbol{\theta_t}\|^2 \right) \\ &= \lambda \boldsymbol{\theta_t} \end{aligned}$$

2. If $1 - y_t \cdot \boldsymbol{x_t^T \theta_t} \geq 0$, then $Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) = 1 - y_t \cdot \boldsymbol{x_t^T \theta_t} + \frac{\lambda}{2} \|\boldsymbol{\theta_t}\|^2$. Then,

$$\begin{aligned} \nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) &= \nabla \left(1 - y_t \cdot \boldsymbol{x_t^T \theta_t} + \frac{\lambda}{2} \|\boldsymbol{\theta_t}\|^2 \right) \\ &= \nabla \left(-y_t \cdot \boldsymbol{x_t^T \theta_t} \right) + \lambda \boldsymbol{\theta_t} \\ &= -y_t \cdot \boldsymbol{x_t} + \lambda \boldsymbol{\theta_t} \end{aligned}$$

where the last step is accomplished by noting that:

$$\begin{aligned} \nabla_\theta \left(\boldsymbol{x^T \theta} \right) &= \nabla_\theta \left(\sum_{i=1}^{|\boldsymbol{x}|} x_i \theta_i \right) \\ &= \left(\frac{\partial}{\partial \theta_1} \left(\sum_{i=1}^{|\boldsymbol{x}|} x_i \theta_i \right), \frac{\partial}{\partial \theta_2} \left(\sum_{i=1}^{|\boldsymbol{x}|} x_i \theta_i \right), \ldots \right) \\ &= (x_1, x_2, \ldots) \\ &= \boldsymbol{x} \end{aligned}$$

Putting these results together, we have that:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha_t \begin{cases} \lambda \boldsymbol{\theta_t} & y_t \cdot \boldsymbol{x_t^T \theta_t} > 1 \\ \lambda \boldsymbol{\theta_t} - y_t \boldsymbol{x_t} & \text{otherwise} \end{cases} \tag{10}$$

We now derive the implicit update for the hinge loss, which is given by:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha_t \nabla Q(\boldsymbol{\theta_{t+1}}, y_t, \boldsymbol{x_t}) \tag{11}$$

The calculations are very similar to the SGD derivation above, and for the two cases depending on the sign of $1 - y_t \cdot \boldsymbol{x_t^T \theta_{t+1}}$ are:

1. If $1 - y_t \cdot \boldsymbol{x_t^T \theta_{t+1}} < 0$, then

$$\nabla Q(\boldsymbol{\theta_{t+1}}, y_t, \boldsymbol{x_t}) = \lambda \boldsymbol{\theta_{t+1}}$$

Then, substituting this result into the implicit update equation above, we can solve to find:

$$\boldsymbol{\theta_{t+1}} = \frac{1}{1 + \lambda \alpha_t} \boldsymbol{\theta_t} \tag{12}$$

2. If $1 - y_t \cdot \boldsymbol{x_t^T \theta_{t+1}} \geq 0$, then

$$\nabla Q(\boldsymbol{\theta_{t+1}}, y_t, \boldsymbol{x_t}) = -y_t \cdot \boldsymbol{x_t} + \lambda \boldsymbol{\theta_{t+1}}$$

Then, substituting this result into the implicit update equation above, we can solve to find:

$$\boldsymbol{\theta_{t+1}} = \frac{1}{1 + \lambda \alpha_t} \left(\boldsymbol{\theta_t} + \alpha_t y_t \cdot \boldsymbol{x_t} \right) \tag{13}$$

Note that during implementation, we should be careful to make sure that we check the sign of $1 - y_t \cdot \boldsymbol{x_t^T \theta_{t+1}}$ after the update, as the derivation of the implicit updates forced an assumption of the sign to begin with. If the assumption was wrong, then the other update function should be used.

Squared-loss. The squared-loss function is given by:

$$L(y, \hat{y}) = (y - \hat{y})^2 \tag{14}$$

The SGD update is given by:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \alpha_t \nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) \tag{15}$$

where

$$Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) = \left(y_t - \boldsymbol{x_t^T \theta_t} \right)^2 + \frac{\lambda}{2} \|\boldsymbol{\theta_t}\|^2 \tag{16}$$

The gradient $\nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t})$ can be calculated as follows:

$$\begin{aligned} \nabla Q(\boldsymbol{\theta_t}; y_t, \boldsymbol{x_t}) &= 2 \left(y_t - \boldsymbol{x_t^T \theta_t} \right) (-\boldsymbol{x_t}) + \lambda \boldsymbol{\theta_t} \\ &= -2(y_t - \boldsymbol{x_t^T \theta_t}) \boldsymbol{x_t} + \lambda \boldsymbol{\theta_t} \end{aligned}$$

So, the SGD update for the log-loss function is:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + 2\alpha_t \left(y_t - \boldsymbol{x_t^T \theta_t} \right) \boldsymbol{x_t} - \alpha_t \lambda \boldsymbol{\theta_t} \tag{17}$$

The implicit update can be derived in much the same way as above. We have the following:

$$
\begin{aligned}
\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} - \alpha_t \nabla Q(\boldsymbol{\theta_{t+1}}, y_t, \boldsymbol{x_t}) \\
\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + 2\alpha_t \left(y_t - \boldsymbol{x_t^T \theta_{t+1}}\right) \boldsymbol{x_t} - \alpha_t \lambda \boldsymbol{\theta_{t+1}} \\
(1 + \alpha_t \lambda)\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + 2\alpha_t \left(y_t - \boldsymbol{x_t^T \theta_{t+1}}\right) \boldsymbol{x_t} \\
(1 + \alpha_t \lambda)\boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t} - 2\alpha_t \boldsymbol{x_t^T \theta_{t+1} x_t} \\
\left[(1 + \alpha_t \lambda) I + 2\alpha_t \boldsymbol{x_t x_t^T}\right] \boldsymbol{\theta_{t+1}} &= \boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t} \\
\boldsymbol{\theta_{t+1}} &= \left[(1 + \alpha_t \lambda) I + 2\alpha_t \boldsymbol{x_t x_t^T}\right]^{-1} (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t})
\end{aligned}
$$

It would be ineffcient to directly compute this matrix inverse and multiplication. We take advantage of sparseness and improve efficiency using the following identity due to Ken Miller:

$$
(G + H)^{-1} = G^{-1} - \frac{1}{1 + tr(HG^{-1})} G^{-1} H G^{-1}
$$

using

$$
\begin{aligned}
G &= (1 + \alpha_t \lambda) I \\
H &= 2\alpha x_t x_t^T \\
g &= tr(HG^{-1}) = tr\left(\frac{1}{1 + \alpha_t \lambda} 2\alpha_t x_t x_t^T\right)
\end{aligned}
$$

so

$$
\begin{aligned}
\boldsymbol{\theta_{t+1}} &= \left[(1 + \alpha_t \lambda) I + 2\alpha_t \boldsymbol{x_t x_t^T}\right]^{-1} (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t}) \\
&= [G + H]^{-1} (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t}) \\
&= \left[G^{-1} - \frac{1}{1 + g} G^{-1} H G^{-1}\right] (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t}) \\
&= \left[G^{-1} - \frac{1}{1 + g} \frac{1}{(1 + \alpha_t \lambda)^2} H\right] (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t}) \\
&= \frac{1}{1 + \alpha_t \lambda} (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t}) - \frac{1}{1 + g} \frac{1}{(1 + \alpha_t \lambda)^2} 2\alpha_t x_t \left(x_t \cdot (\boldsymbol{\theta_t} + 2\alpha_t y_t \boldsymbol{x_t})\right)
\end{aligned}
$$

(b) We downloaded and compiled Bottou's SGD package.

(c) See `svmimplicit.cpp`.

(d) Here we report the results from running the SGD experiment. We compare performance of baseline SGD, ASGD, and our implicit SGD for each of the log loss and hinge loss functions and each of the RCV1 and alpha datasets across three metrics: training time, test (misclassification) error, and cost, measured as

$$
C = L + 0.5\lambda ||\Theta||^2
$$

where L denotes the average loss over all test runs and Θ the parameter vector. (We leave out LibLinear because we could not format the data correctly in time.) We ran all the experiments on Andrew's personal computer using one core, and the longest run in terms of computation time that we tried was

approximately 20 seconds. Each run was done individually from the command line (instead of packaging all runs into one script) for easier debugging and due to the small number of runs necessary.

Unless otherwise stated, SGD algorithms use their default regularization parameters (close to $\lambda = 10^{-5}$) and learning rate $\eta_t = \frac{\eta_0}{1+\lambda\eta_0 t}$ as specified in the Bottou SVM README, with η_0 picked by the Bottou package unless otherwise specified. We chose these regularization parameters because they optimized the performance of each run.

1. RCV1 Benchmark, hinge-loss.

algorithm	training time (s)	test error (%)	cost
SGD	0.87	6.005	0.244
ASGD	0.86	6.018	0.244
Implicit	0.71	5.184	0.144

2. RCV1 Benchmark, log-loss.

algorithm	training time (s)	test error (%)	cost
SGD	3.47	5.175	0.153
ASGD	4.03	5.145	0.154
Implicit ($\eta_0 = 10$, not 4)	1.92	5.262	0.537

   For implicit SGD, I use learning rate $\eta_t = \frac{\eta_0}{1+\lambda\eta_0 t/2}$.

3. Alpha dataset, hinge-loss.

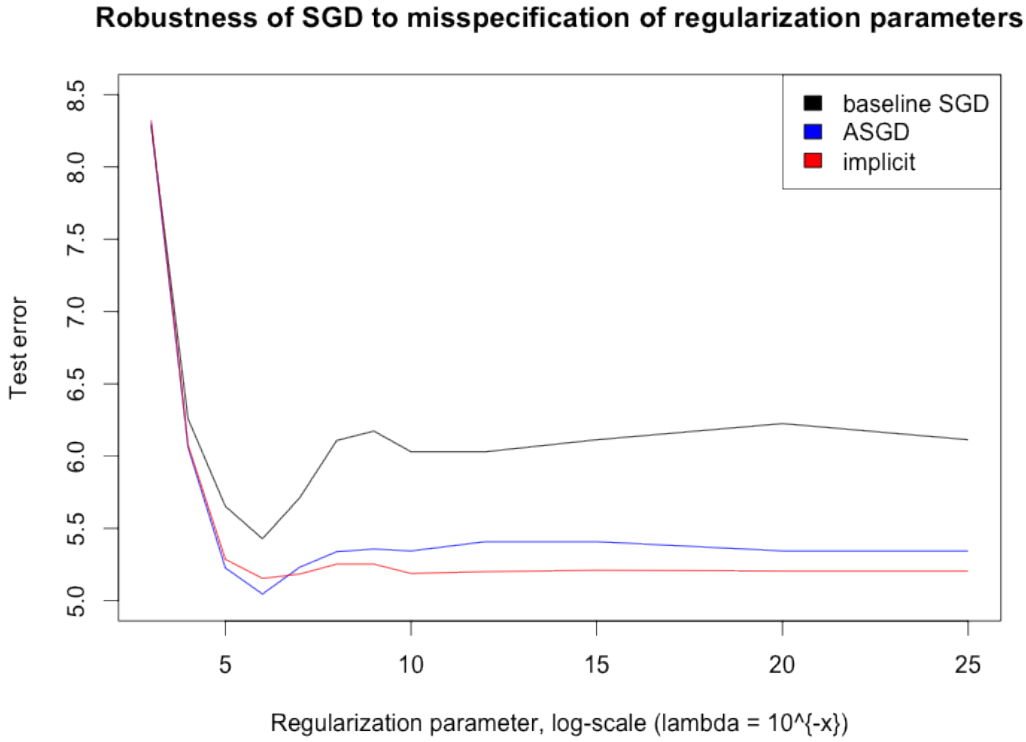
algorithm	training time (s)	test error (%)	cost
SGD	3.37	22.7	0.548
ASGD	2.27	21.83	0.532
Implicit ($\eta_0 = 10$, not 0.25)	2.29	22.18	0.565

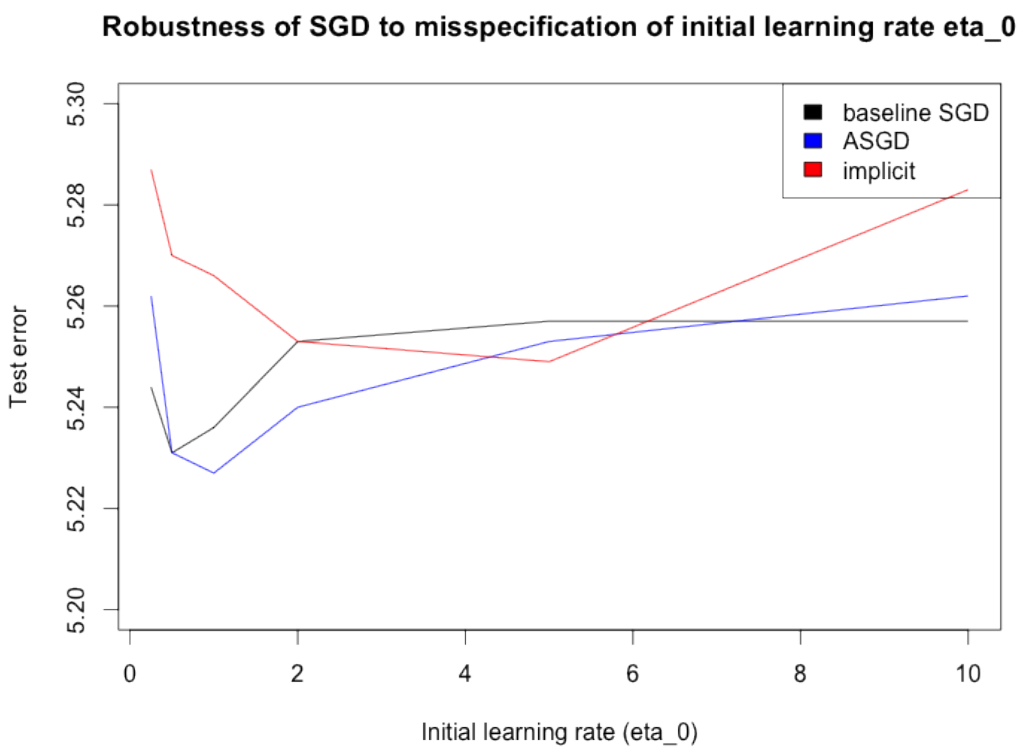
   Implicit SGD here was improved by using default $\lambda = 10^{-5}$ (whereas SGD and ASGD used $\lambda = 10^{-6}$).

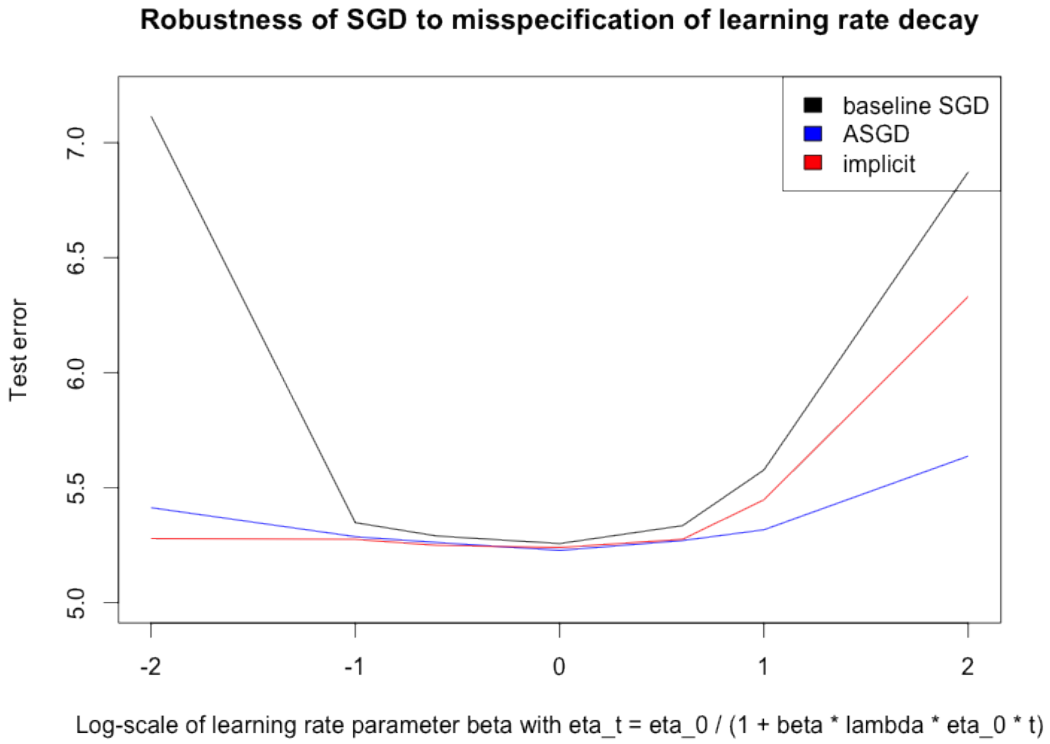
4. Alpha dataset, log-loss.

algorithm	training time (s)	test error (%)	cost
SGD	9.31	22.11	0.477
ASGD	3.5	21.9	0.474
Implicit ($\eta_0 = 10$, not 0.25)	3.7	21.9	0.542

We also wanted to test the robustness of each method to misspecification of parameters such as regularization rate and learning rate. Looking at our results, implicit SGD is only noticeably performing better (considering training time and test error) in hinge-loss on the RCV1 dataset, so we considered each algorithm's performance while varying its regularization rate. By taking the test error at the end of 1 second of training, we find that implicit SGD is slightly more robust to changes in regularization rate than the other two:


Robustness of SGD to misspecification of regularization parameters
baseline SGD
ASGD
implicit
Test error
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.5
5
10
15
20
25
Regularization parameter, log-scale (lambda = 10^{-x})


Robustness of SGD to misspecification of initial learning rate eta_0
baseline SGD
ASGD
implicit
Test error
5.20
5.22
5.24
5.26
5.28
5.30
0
2
4
6
8
10
Initial learning rate (eta_0)



4 Discussion

Our first main result is that implicit SGD only outperformed baseline SGD and ASGD in both training time and test error in the hinge-loss case for the RCV1 dataset while doing as well or worse than ASGD in the other cases (and always better than baseline SGD). In hinge-loss for RCV1, implicit SGD converged slightly more quickly and with less error, and in log-loss, it converged more quickly than the other implementations but with slightly more error. In alpha, it converged at approximately the same rate as ASGD (and faster than baseline SGD) and had error on par with ASGD, and better than baseline SGD.

We hypothesized that our implicit SGD implementation would (a) have lower error than other implementations, fixing convergence time, and (b) have lower convergence time, fixing error. That this did not happen is likely not due to incorrect derivation equations but could follow from bugs in our implementation, since we checked our implicit SGD update equations with Panos. However, our implementations' performance is quite close to the SGD and ASGD from the Bottou package, which suggests that the update equations were coded correctly. A more likely source of this underperformance is the tradeoff between making our code faster and computing things exactly; for instance, the `FVector combine` function used for adding was more exact but slower. Another possible source of underperformance was our inability to account for nonzero bias in our classifier, which required us to set the bias for all our classifiers to be zero (including baseline SGD and ASGD). This restriction may have affected implicit SGD disproportionately.

Our second main result is that implicit SGD is more robust than both other SGDs to misspecifications in regularization coefficient, and more robust than baseline SGD in misspecifications in learning rates. First, analyzing the graph of regularization parameter versus test error, we see that implicit SGD achieves a dramatically lower test error than baseline SGD for values of $\lambda = 10^{-i}$ for $i \geq 4$ (which happens to be close to Bottou's recommended regularization coefficient $\lambda = 10^{-5}$) and a slightly lower test error than ASGD. Next, we analyzed the sensitivity of each implementation to initial learning rates η_0, finding no significant

differences between each implementation's robustness while varying η_0 (all test errors were still confined to $(5.22, 5.29)$ after taking the implementation test errors by 1 second of training time). However, we did find that our implicit SGD was more robust than baseline SGD to different rates of decay β in the learning rate equation

$$\eta_t = \frac{\eta_0}{1 + \beta\lambda\eta_0 t}$$

but ASGD is even more robust than our implicit SGD. This might be due to implicit SGD needing higher learning rates overall, so that the ranges of values on which implicit SGD versus ASGD are more robust differ.

5 Contributions

On the implementation side, Adam did a majority of the work, being the first to download Bottou's SGD package and get it working (1.4b), and implementing each of the loss functions with implicit update (1.4c), as well as working with Brandon to figure out some of the math associated with those updates. He also debugged the `svmimplicit.cpp` code and did a little tuning of the parameters.

On the analysis side, Brandon derived the SGD and implicit updates for the three loss functions (1.4a). Jason gave the intuition for Sakrison's method and the implicit method computing the MLE (1.1), and performed the derivations of the desired properties of GLMs (1.2).

Finishing off the implementation, Andrew ran Adam's implicit SGD, along with baseline SGD and AGSD, on RCV1 and alpha datasets, collecting training time, test error, and cost data (1.4d, e). He tried but failed to get LibLinear working. He also did a little work to compile each team member's tex into this writeup and wrote the results and discussion sections (including the plots and data tables).

Adam and Brandon made large contributions in terms of keeping the team going forward, communicating their progress to the group and ensuring the group stayed knowledgeable, and starting on their work early so that the rest of the team had an easier time finishing off the missing pieces at the end.