CS690OP midterm

March 7, 2016

Question 1 Rosenbrock's Function

a Steepest Descent

Figure 1 shows the convergence of the method of steepest descent for a variety of different starting points and step sizes. Steepest descent doesn't usually converge unless the step size is very small (< 0.0014 in my rough tests), and takes a few thousand iterations to do so.

b Newton's Method

Figure 2 shows the convergence rate of Newton's Method. This method converges almost instantly (within 6 or 7 steps), no matter where we start from.

Question 2 Subgradients

a Subgradient of Max

this

b Projected Subgradient

To project onto the line segment, we can first project onto the line $x_1 + x_2 = 1$. If $x_1, x_2 > 0$, then we have the projection onto the line segment. If $x_1 < 0$, we can use the point $(0, 1)$. Otherwise, we can use the point $(1, 0)$.

Here's a proof that this method of projection works for the $L - 2$ norm. Since we are just trying to keep the gradient descent in the feasible region, we can use any norm that is convenient.

Let $\mathbf{proj}_L x$ be the projection of a point x onto the line $x_1 + x_2 = 1$. Suppose our method projected the point onto p_1 which has a distance d_1, but the actual nearest was p_2 with a distance $d_2 < d_1$. Let $\Delta_1 = \|p_1 - \mathbf{proj}_L x\|_2$, let $\Delta_2 = \|p_2 - \mathbf{proj}_L x\|_2$, and let $h = \|x - \mathbf{proj}_L x\|_2$. Since we pick the nearest point to $\mathbf{proj}_L x$ on the line segment, $\epsilon = \Delta_2 - \Delta_1 \geq 0$. By the pythagorean theorem,

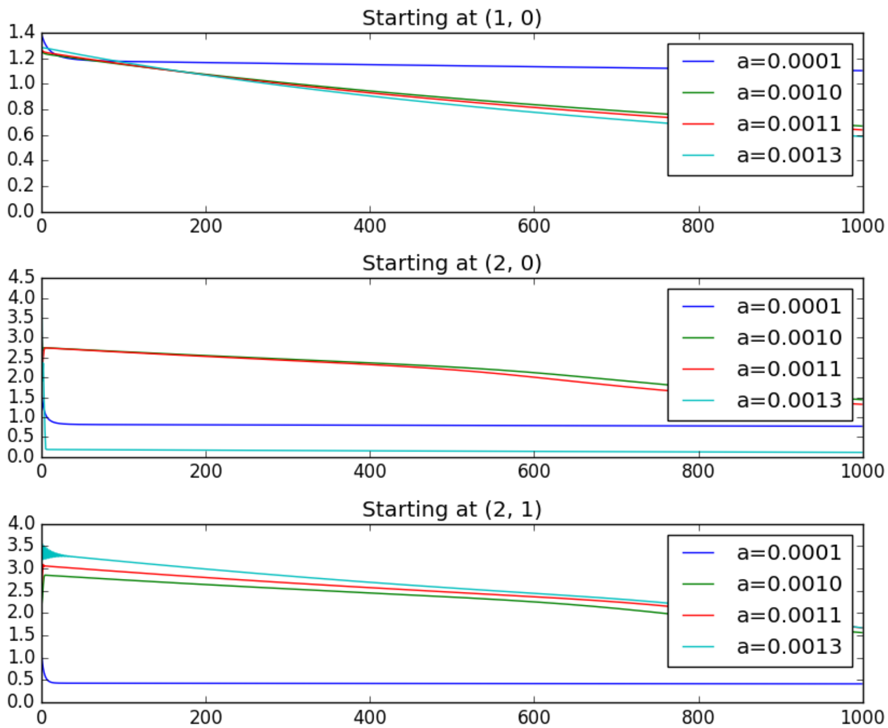

Figure 1: Convergence of Steepest Descent, for various starting points and step sizes

$d_1^2 = \Delta_1^2 + h^2$ and $d_2^2 = \Delta_1^2 + h^2 = (\Delta_1 + \epsilon)^2 + h^2 = \Delta_1^2 + 2\Delta_1\epsilon + \epsilon^2 + h^2 \geq d_1$, which is a contradiction.

Starting from $(0, 1)$, the gradient step takes us to $(-1, -1)$, which is projected to $(0.5, 0.5)$. The next gradient step takes us to $(-1.5, 1)$, which is projected to $(1, 0)$ which is optimal.

Question 3 Kaczmarz

Figure 4 shows the convergence of the different kaczmarz methods for $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ and b picked as a random number in the range $[0, 1)$.

For the same A and $b = \begin{bmatrix} 0.3467 & 0.8979 & 0.9461 \end{bmatrix}^T$, the distal method

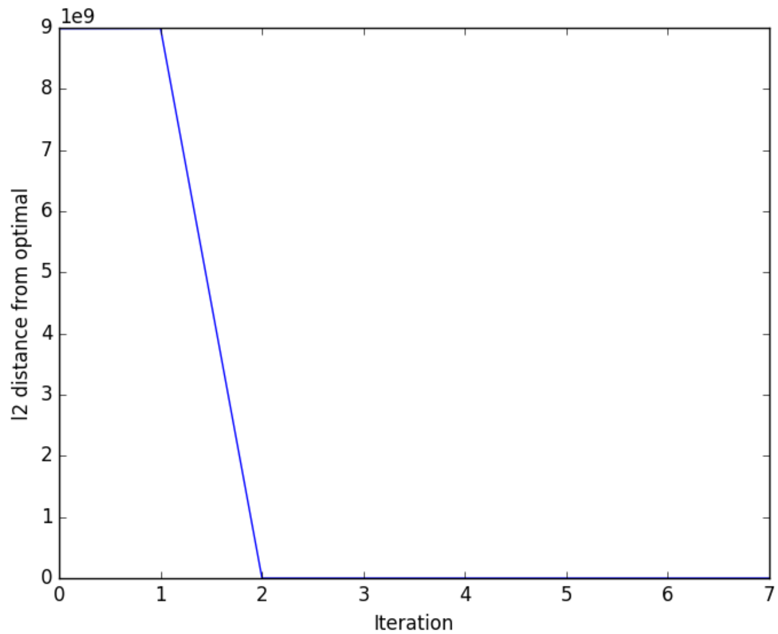

Figure 2: Convergence of Newton's Method on the Rosenbrock Function, starting from $(-52970, -2159)$

outperforms all the others. The distal method converges in 14 iterations, while the random method converges in 28, and the cyclic method takes > 2000.

Question 4 Oblique Projections

a Example 1

$w_{\text{best}} = \frac{1}{5}r_1 + \frac{2+\gamma}{5(1-\gamma)}r_2$ $w_{\text{TD}} = \frac{r_1+2r_2}{5-6\gamma}$ $w_{\text{BR}} = \frac{(1-2\gamma)r_1+(2-2\gamma)r_2}{(1-2\gamma)^2+(2-2\gamma)^2}$

I did the omitted algebraic steps to find $\frac{e(w_X)}{e(w_{\text{best}})}$ for both TD and BR.

For TD, $\frac{e(w_{\text{TD}})}{e(w_{\text{best}})} = \frac{5(5-12\gamma+9\gamma^2)}{(5-6\gamma)^2}$. This confirms that the TD error ratio is independent of r_1 and r_2. As γ approaches $\frac{5}{6}$, the denominator of the TD error ratio approaches 0, and the error ratio approaches infinity.

For BR, $\frac{e(w_{\text{BR}})}{e(w_{\text{best}})} = \frac{5\left(16g^4-40g^3+45g^2-24g+5\right)}{(8g^2-12g+5)^2}$. This confirms that the BR error ratio is also independent of r_1 and r_2. As γ approaches $\frac{5}{6}$, the denominator of this error ratio approaches $\frac{25}{81}$, so the error ratio is bounded. In fact, over the interval $\gamma = (0, 1)$, the error ratio is always below 14.

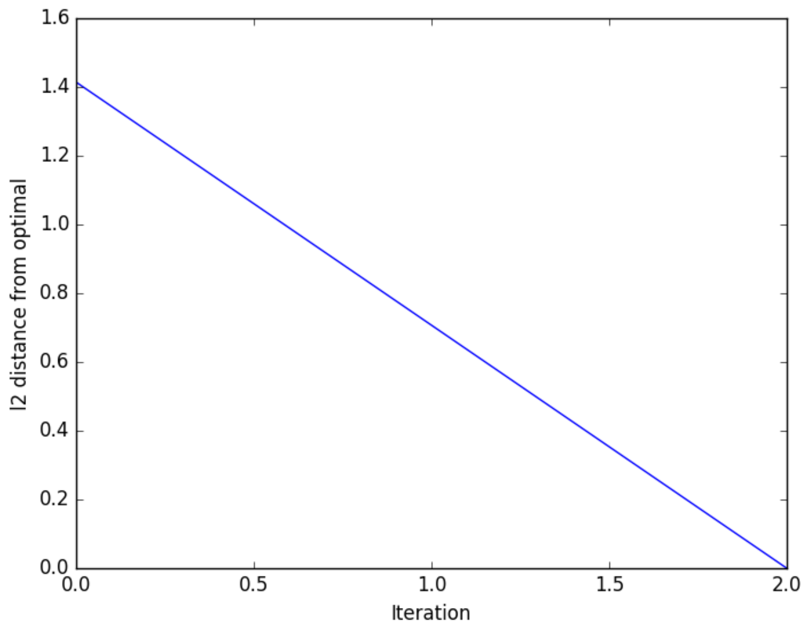

Figure 3: Convergence of Projected Gradient Descent

b How the methods use oblique projections

Both methods estimate the value of v with a linear combination of features ϕ. They do this by obliquely projecting v onto $\mathbf{span}(\Phi)$. The resulting vector $\hat{v}$ is a linear combination of the different features which is nearest to v in some sense. Instead of doing the orthogonal projection onto $\mathbf{span}(\Phi)$, both TD and BR find the vector on $\mathbf{span}(\Phi)$ which is orthogonal to some other surface $\mathbf{span}(L^T X)$.

The value of X is different for each method. Let Ξ be a diagonal matrix of a probability distribution over the states of the system, r be a vector of rewards, and L be a matrix where $v = L^{-1}r$. TD finds the vector orthogonal to $\mathbf{span}(L^T \Xi \Phi)$, while BR finds the vector orthogonal to $\mathbf{span}(L^T \Xi L \Phi)$.

For example 1 in the paper, we have the following values for the variables:

$$P = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{1}$$

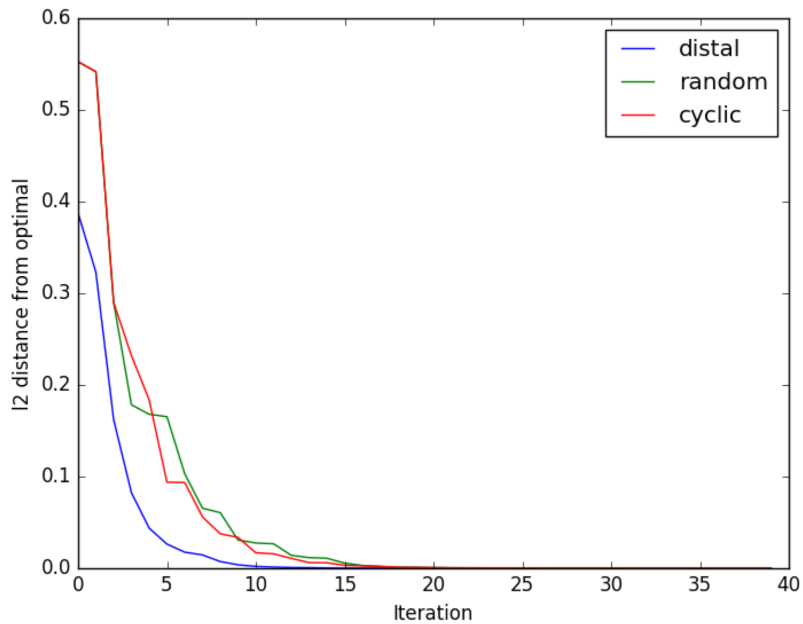

Figure 4: Convergence of different kaczmarz methods for a 3x3 A and a random starting b

$$L = \begin{bmatrix} 1 & 0 \\ -\gamma & 1-\gamma \end{bmatrix} \tag{2}$$

$$\Xi = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \tag{3}$$

$$\Phi = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{4}$$

This means we have the following values for X_{TD} and X_{BR}, which suggests that $X_{BR} = X_{TD} - \vec{\gamma}$

$$X_{TD} = \Xi\Phi \tag{5}$$

$$= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{6}$$

$$= \begin{bmatrix} 0.5 \\ 1 \end{bmatrix} \tag{7}$$

$$X_{BR} = \Xi L \Phi \tag{8}$$

$$= \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\gamma & 1-\gamma \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \tag{9}$$

$$= \begin{bmatrix} 0.5-\gamma \\ 1-\gamma \end{bmatrix} \tag{10}$$

Question 5 Matrix Completion

See Table 2 for the RMSE for the different algorithms, Figure 5 for the convergence of the Factorization algorithm, and Figure 6 for the convergence of the SVT algorithm.

I implemented the three baseline methods and the three more advanced methods in python using numpy. See MOVIELENS/METHODS.PY for the implementation. A list of the parameters I used is in Table 1.

For the factorization algorithm, I used a method from [1] (sec. 5) instead of the provided method, which appears to be an equivalent version of gradient descent. The paper's formulation allowed me to write one update step per iteration as a few matrix operations instead of trying to do one update per row per iteration. This made things much, much faster.

Table 1: Parameter Values

Algorithm	Variable	Value
Mixture Mean	α_1	0.452
SVT	τ	1
SVT	δ_q	1.9
Factorization	α	0.0000015

Table 2: RMSE for various Matrix Completion algorithms

Algorithm	Part 1	Part 2	Part 3	Part 4	Part 5	Average
Factorization[1]	3.4874	3.4448	3.4226	3.4310	3.4578	3.4487
Global Mean	1.1233	1.1279	1.1097	1.1104	1.1149	1.1172
Mixture Mean	1.8956	1.8953	1.9136	1.9101	1.9181	1.9066
Movie Mean	0.9816	0.9790	0.9807	0.9761	0.9857	0.9806
SVT[1]	3.6986	3.6958	3.6712	3.6693	3.6735	3.6817
User Mean	3.7349	3.7304	3.7632	3.7307	3.7887	3.7496

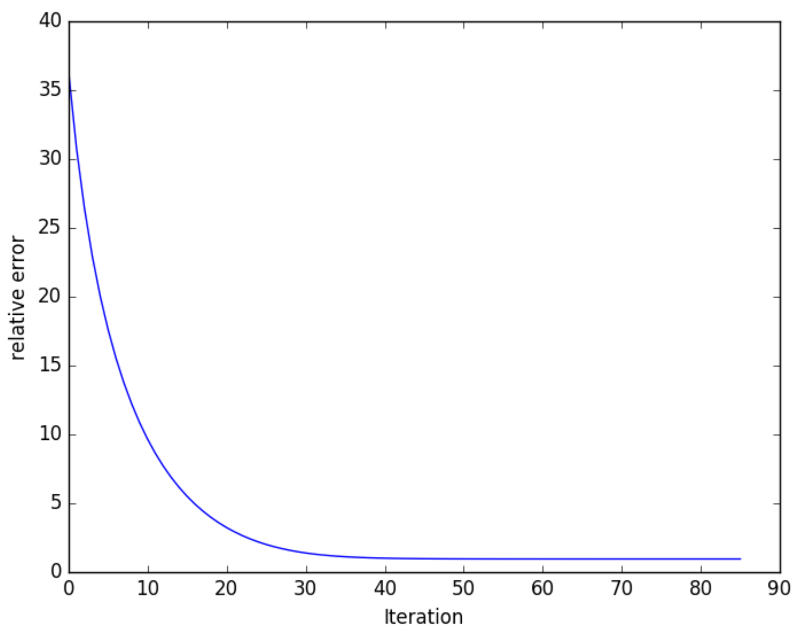

Figure 5: Convergence of the factorization algorithm for 100k entries

[1]These algorithms were only run on the 100k dataset because they were incredibly slow on the 1M dataset

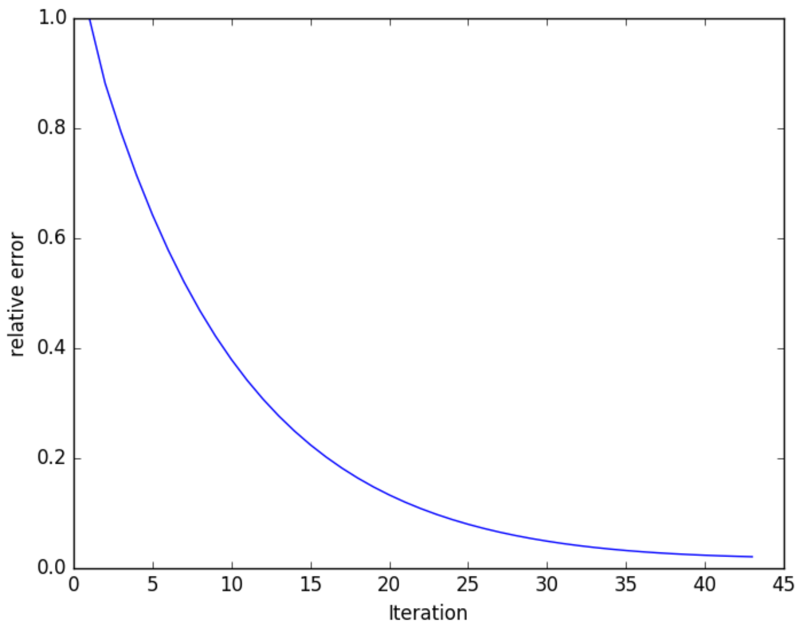

Figure 6: Convergence of the SVT algorithm for 100k entries

References

[1] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *In NIPS*, pages 556–562. MIT Press, 2000.