1: Classify the hand writing digits in extracted features

Data: The data set is of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990). The task here is to classify the hand writing digits. I utilize the two features that are provided by the data set to separating digit ”1” and the rest.

Model: Linear least squares, logistic regression and the PLA model are used to do the classification.

Linear regression: In the training stage, the model is

$$\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|^2,$$

where $\mathbf{A}$ and $\mathbf{b}$ are defined below.

Since the featured data includes two features, ”intensity” and ”symmetry”, and 7291 training samples, the data matrix $\tilde{\mathbf{A}}$ is of size 7291$\times$ 2. For any given data point $\mathbf{a}$, the linear prediction model produces the above following form

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{a} + x_0,$$

where x_0 is the variable that characterizes the ”intercept” of the line. Therefore, the following classifier is needed for the problem

$$x_1a_1 + x_2a_2 + x_0 = 0,$$

which precisely defines a line on a 2-dimensional plane. Therefore, another decision variable is needed, so the final variable vector is $\mathbf{x} = [x_1, x_2, x_3]^T \in \mathbb{R}^3$, and the final data matrix is

$$\mathbf{A} = [\tilde{\mathbf{A}}, \mathbf{1}] \in \mathbb{R}^{7291\times 3},$$

where $\mathbf{1}$ is the all 1 column vector of size 7291.

Since the task is to classify digit ”1” and the rest, for all instances i correspond to the letter 1, the label b_i is +1, and for the rest of the instances, the label is -1.

Logistic regression: The same approach as the above can be done for the logistic regression. 3-dimensional variables, augmenting the data matrices, constructing the labels $\mathbf{b}$ are still needed.

The objective function for logistic regression is

$$L(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m} \ln(1 + e^{-b_i\mathbf{a}_i^T\mathbf{x}}).$$

PLA: A direct application of the PLA algorithm is not going to work due to the fact that the real-data is not linear separable. The modifed PLA method has to be used, which checks the decrease of the objective function $L(\mathbf{x})$ every iteration

$$L(\mathbf{x}) = \sum_{i=1}^{m} \max\{-b_i\mathbf{a}^T x_i, 0\}.$$

Algorithm: For linear least squares, several first order algorithms are used, including gradient decent method with constant step size and Armijo rule, and block coordinate descent method. For logistic regression, gradient descent algorithm with constant step size is used. For the PLA model, the pocket algorithm is implemented.

Gradient descent: To solve the problem, a family of gradient descent methods of the following form is implemented,

$$\mathbf{x}^{r+1} = \mathbf{x}^r + \alpha_r \mathbf{d}^r,$$

where $r = 0, 1, \cdots$. A few different choices of stepsize are tried:

- Constant stepsize, $\alpha = \frac{1}{\lambda_{max}(\mathbf{A}^T\mathbf{A})}$.
- Armijo stepsiz selection rule: Let $\mathbf{d}^r = -\nabla f(\mathbf{x}^r)$, $\sigma \in (0, \frac{1}{2})$, s and $0 < \beta < 1$. α keeps shrinking by $s, \beta s, \beta^2 s, \cdots$ until the following is satisfied,

  $$f(\mathbf{x}^r + \alpha \mathbf{d}^r) - f(\mathbf{x}^r) \le \sigma\alpha\langle \nabla f(\mathbf{x}^r), \mathbf{d}^r \rangle.$$

  Parameters are randomly generated for each stepsizes.

Coordinate descent: The usual regression problem can be rewritten as

$$\min_{\mathbf{x}} \frac{1}{2} \| \sum_{i=1}^{n} \mathbf{a}_i x_i - \mathbf{b} \|^2,$$

where $\mathbf{a}_i$ represents ith column of $\mathbf{A}$, x_i is the ith element of $\mathbf{x}$. Minimizing with ith coordinate, we have

$$\min_{x_i} \frac{1}{2} \|\mathbf{a}_i x_i + \sum_{j \neq i}^{n} \mathbf{a}_j x_j^r - \mathbf{b}\|^2.$$

Or for the ith subproblem

$$\min_{x_i} \frac{1}{2} \|\mathbf{a}_i x_i + \sum_{j \neq i}^{n} \mathbf{a}_j x_j^r - \mathbf{b}\|^2,$$

the gradient with respect to x_i is $\mathbf{a}_i^T(\mathbf{a}_i x_i^r + \sum_{j \neq i}^{n} \mathbf{a}_j x_j^r - \mathbf{b})$. The Hessian for the ith subproblem is a scalar,

$$L_i = \mathbf{a}^T a_i > 0.$$

Both ways give solution or update,

$$x_i^{r+1} = \frac{1}{\mathbf{a}_i^T \mathbf{a}_i} \mathbf{a}_i^T (\mathbf{b} - \sum_{j \neq i}^{n} \mathbf{a}_j x_j^r).$$

Different rules for picking coordinates are tried with different block sizes, including cyclic, randomized, and permuted fashions.

Pocket algorithm: The objective function is

$$L(\mathbf{x}) = \sum_{i}^{m} \max\{-b_i \mathbf{x}^T \mathbf{a}_i, 0\}.$$

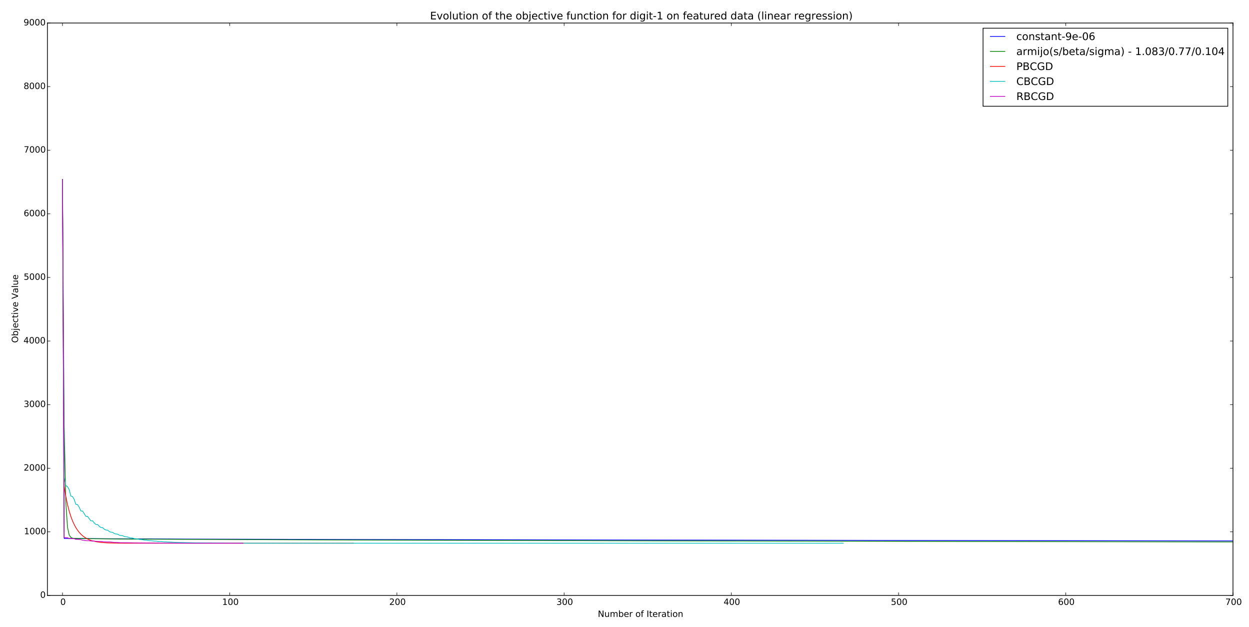

Figure 1: Objective value reduction of algorithms for linear regression

For classify the training data, the model **sgn**($\mathbf{a}^T\mathbf{x}$) is to be learned. Starting at an arbitrary solution $\mathbf{x}$, a misclassified point is picked: **sgn**($\mathbf{a}^T\mathbf{x} \neq b_n$) (or equivalently, $b_n\mathbf{x}^T\mathbf{a}_n < 0$). The weight vector is updated by $\mathbf{x} \leftarrow \mathbf{x} + b_n\mathbf{a}_n$ (the gradient is $\nabla g_i((x) = -b_n\mathbf{a}_i$ and the gradient is zero for each correct data point i). The previous algorithm is run for one iteration to obtain $\mathbf{x}^{r+1}$. And $L(\mathbf{x}^{r+1})$ is evaluated; if $\mathbf{x}^{r+1}$ is better than previous solution in terms of the loss, $\mathbf{x}^{r+1}$ is the best solution so far. This way, the loss function is decreasing.

Result: The results are shown in the figures below. Fig. 8 shows objective reduction of different algorithms for the models. The first two lines result from gradient descent methods with a constant step size and an Armijo rule. PBCGD, CBCGD, and RBCGD are permuted, cyclic, and randomized block coordinate descent method, respectively. In this experiment, PBCGD has a block of size 3, CBCGD 1, and RBCGD 2. The coordinate descent algorithms all converged but at different rates (when the objective does not change). RBCGD converged at iteration 109, PBCGD at 175 and CBCGD at 468. The gradient descent methods reduce objective but not reach to the optimality within given iteration limitation (700 in this case). All the algorithms start at the same arbitrary solution. As for logistic model, in Fig. 9, the gradient descent method reduce objective value slowly in contrast. It does not terminate before the iteration limitation. As presented in Fig. 3, for the modified PLA model solved by the pocket algorithm, the objective is reduced, although not at each iteration, which is the feature of the pocket algorithm on non-linear separable problems.

The lines of the classifiers are drawn in Fig. 10 represent the separation lines from the best solutions obtained by different model/algorithm pairs. Among them, the modified PLA model gives the best testing error rate (1.99%), followed by a rate of 2.24% obtained by linear regression with constant step size and Armijo rule gradient descent method, and RBCGD. PBCGD and CBCGD perform slightly worse at an error rate of 2.29%. The worst performance is given by the logistic regression with gradient descent method with the rate around 86.85%.

Analysis: In the results, the modified PLA model solved by the pocket algorithm perform well in terms of the testing error rate, and the objective reduces quite fast. This suggests for this task, separating digit 1

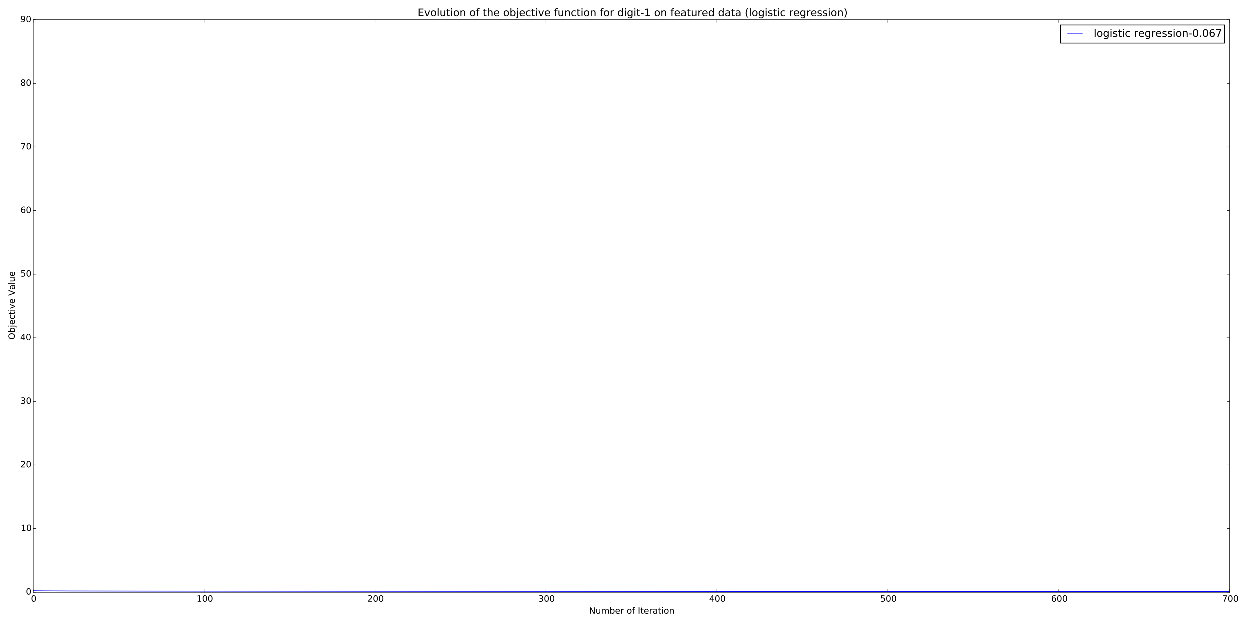

Figure 2: Objective value reduction of algorithm for logistic regression

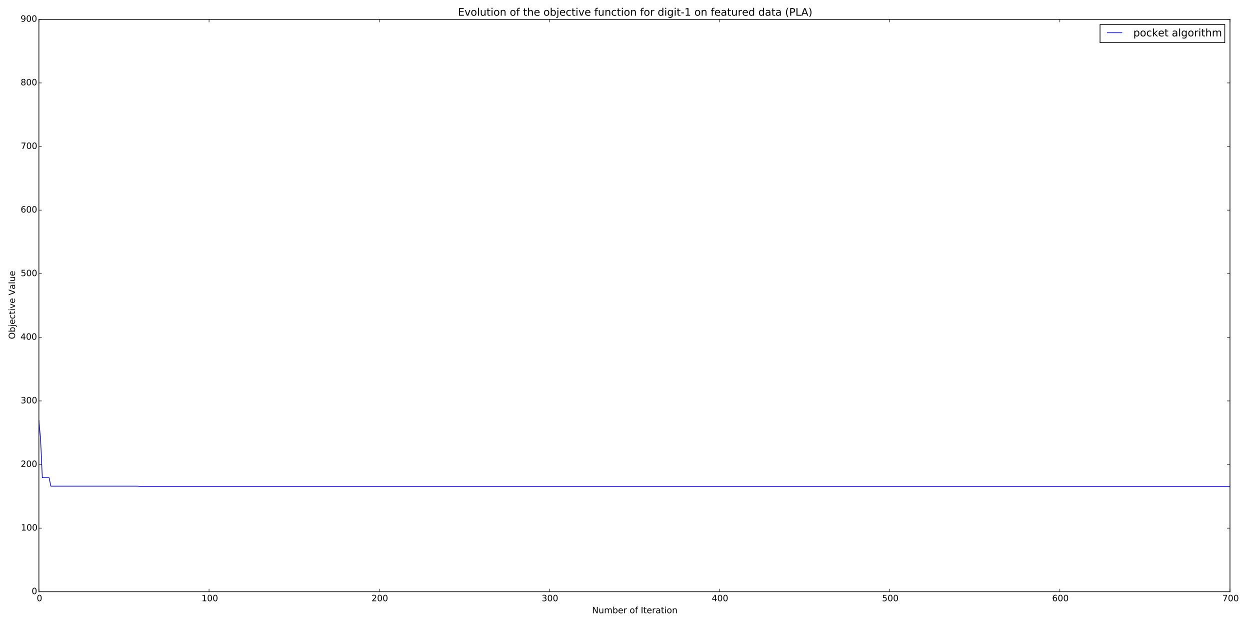

Figure 3: Objective value reduction of algorithm for PLA model

from the rest, the objective function of the modified model works better than the other interested models. The logistic regression model solved by the gradient descent algorithm with a constant step size does not obtain a result anywhere near the acceptable. The reason may be the objective decreases too slow so a better solution cannot be reached within the given iteration number. Overall, the linear regression models work fine with any of the implemented algorithms. The error rates are around 2%. However, coordinate descent methods perform better than the gradient descent methods in terms of convergence speed (arriving at the optimality faster).

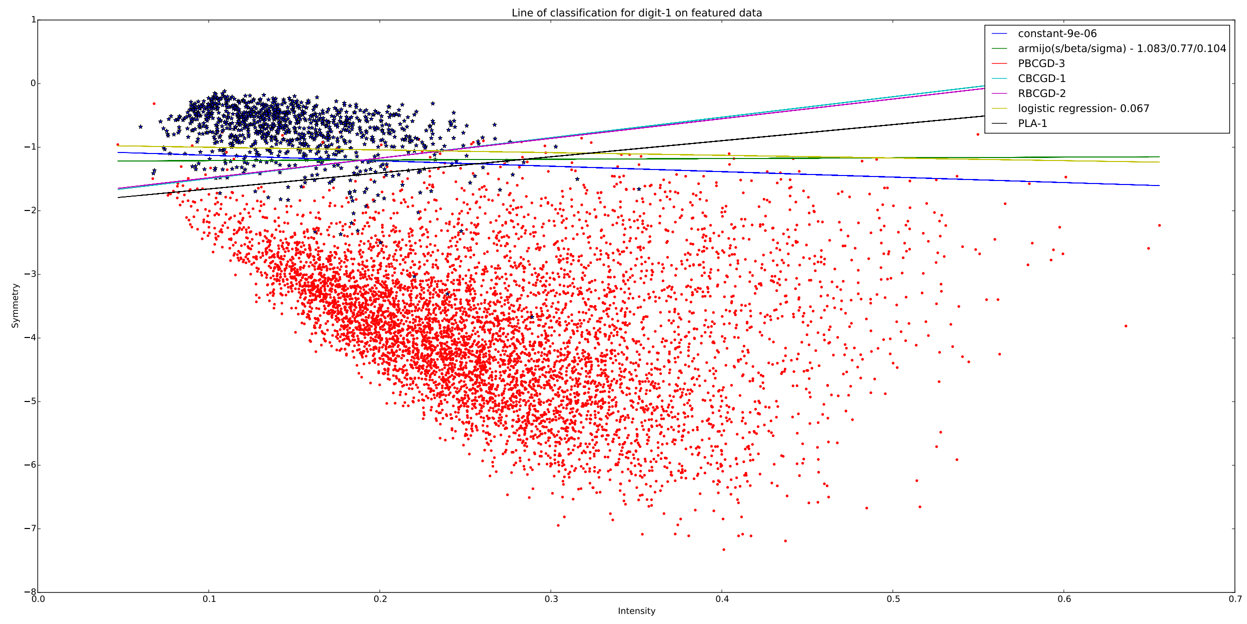

Figure 4: Classifiers of different models

2: Classify the hand writing digits in original description

Data: The data set is of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. The original scanned digits are binary and of different sizes and orientations; the images here have been deslanted and size normalized, resulting in 16 x 16 grayscale images (Le Cun et al., 1990). The task here is to classify the hand writing digit. Similar tasks as in the previous problem are performed, but using the original data files. Here as extracted features are no longer available, much higher dimension data has to be dealt with.

Model: The exact same models used in the previous problem are applied to the original data files. The original description of the digits are dealt with to separate digit ”1” from the rest.

Algorithm: The same algorithms described in the previous problem are applied to solve the models. Here the coordinate descent methods have the same block size of 135.

Result: Fig. 11 shows results from 5 algorithms for the linear regression model. PBCGD does not reach a objective value below 9000 within given maximum iteration (700), so the line does not appear in the figure. RBCGD is able to obtain the smallest objective value at the end of the run, but the other coordinate descent method, CBCGD, perform only better than PBCGD but much worse than the two gradient descent methods. The armijo rule present the second best performance, followed by the constant size based gradient descent method. As for the logistic regression model in Fig. 12, again, the gradient algorithm with a constant step size does not reduce the objective value comparably fast; in contrast, the pocket algorithm works for modified PLA model in a way that the objective can be reduced fast in the earlier runs as shown in Fig. 13.

With the original data, the PLA model gives the best testing error at 1.09%, followed by RBCGD at 2.84%. The other two coordinate descent methods have much higher error rates at 28.75% and 15.89%. In

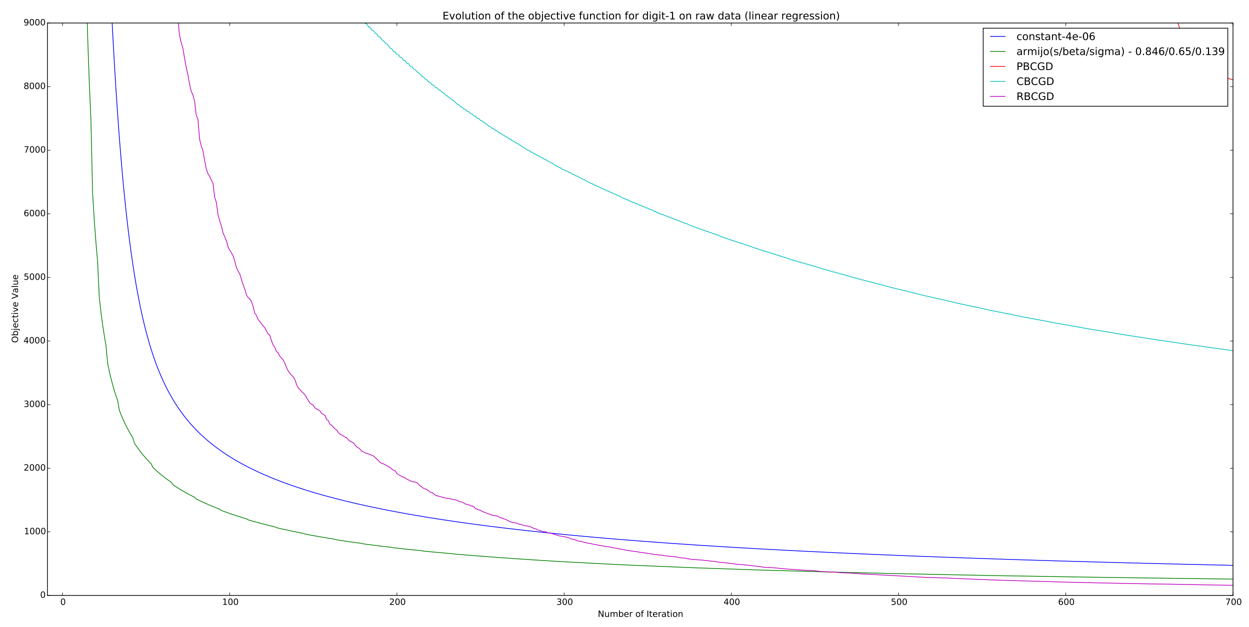

Figure 5: Objective value reduction of algorithms for linear regression

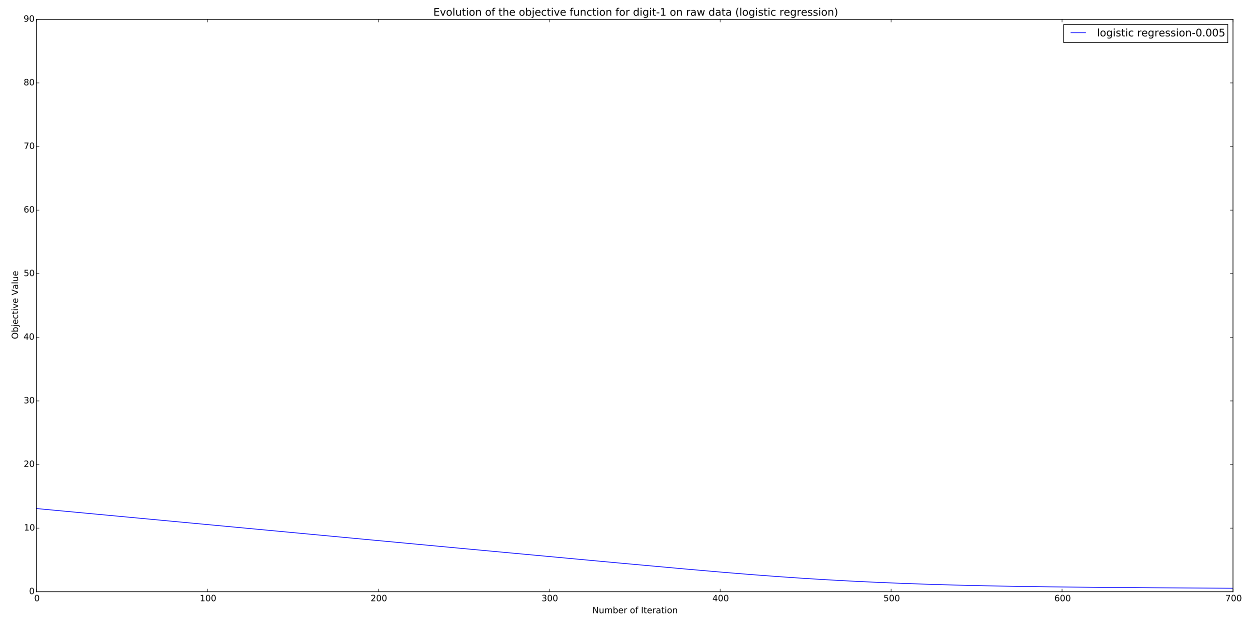

Figure 6: Objective value reduction of algorithm for logistic regression

contrast, the gradient descent methods with constant and Armijo rule step sizes present error rates at 6.68% and 2.49%, respectively. Again, within the limited iteration, the constant step size gradient descent method on linear regression perform unfavorably.

Analysis: When it comes to doing training on the raw data, the modified PLA model with the pocket algorithm works surprisingly at an error rate of 1.05% and again the objective drops steeply in the early iterations. The objective actually reaches zero during the run time. And the logistic regression model tends

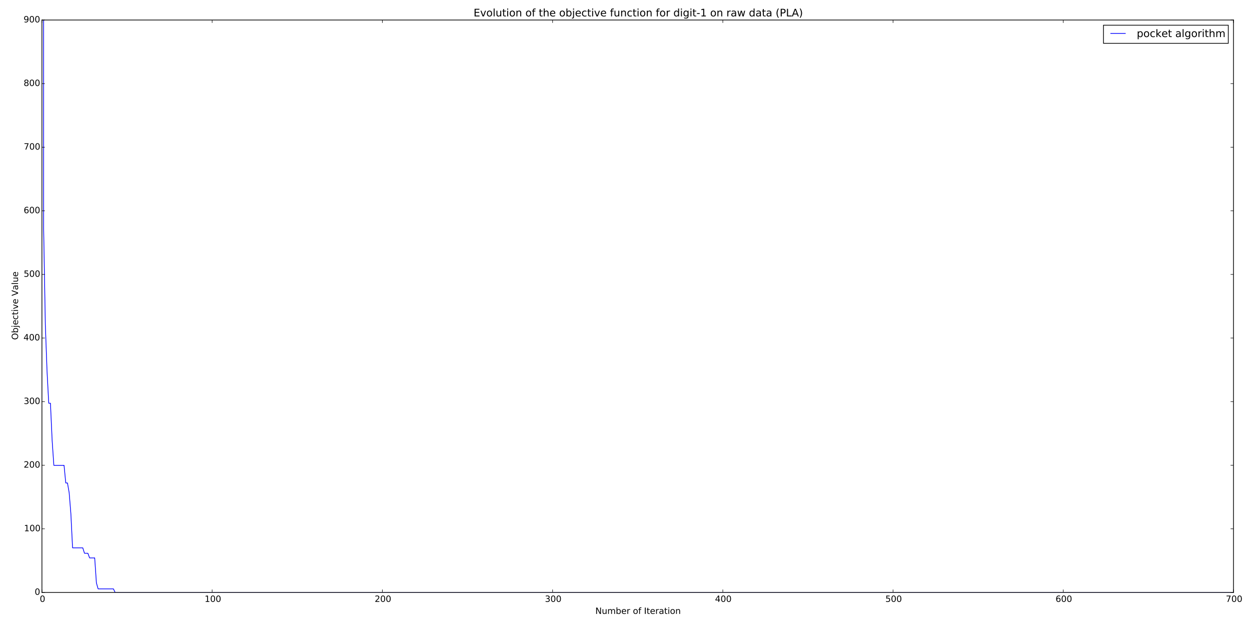

Figure 7: Objective value reduction of algorithms for PLA model

to perform worst in both data sets (feature and original); although the objective decreases, the slope of the reduction rate approaches 0, and the testing error is more than 86%. RBCGD performs the second best with a little bit higher error rate than it does on the feature data. Randomized the selection of block seems to have a better performance than the other two coordinate descent methods, including PBCGD, and CBCGD. Between the gradient algorithms, the Armijo rule based one tends to have better performance in terms of objective reductions and hence error rates; due to the large dimensions, generally all the models and algorithms have slight degradation in error rate and objective value, except the PLA model, which have an improved error rate. One thing is worth noticing is that the block coordinate descent methods take more computation time than other algorithms in the experiment. The reason I think is because instead of taking the entire solution vector, the computation is on a subset of the vector based on predefined block size at each iteraition.

More tasks are performed with the objective of classifying digit ”2” and the rest. The results are shown in figures as follows for comparison purpose.

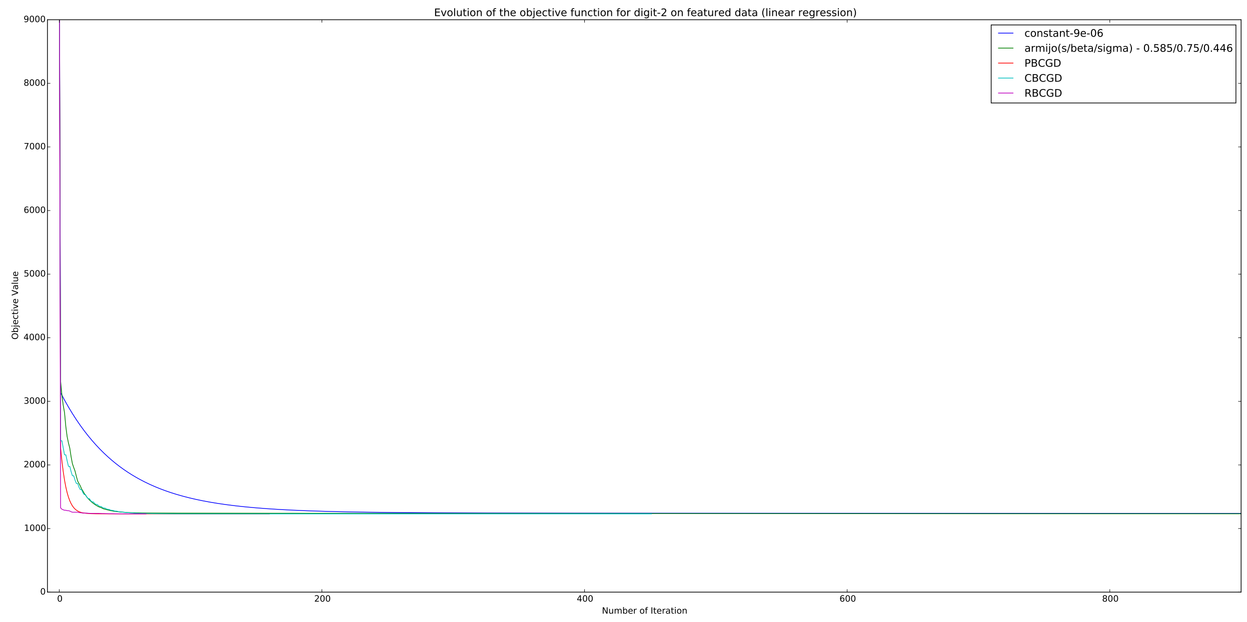

Figure 8: Objective value reduction of linear regression for digit 2 on raw data

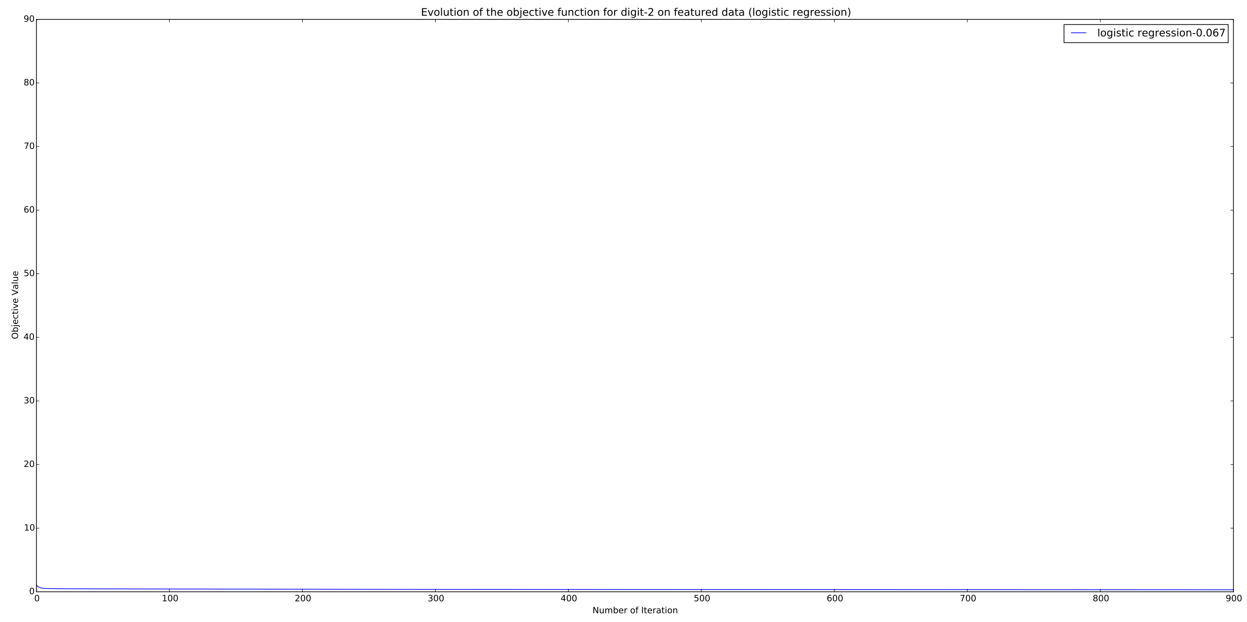

Figure 9: Objective value reduction of logistic regression for digit 2 on raw data

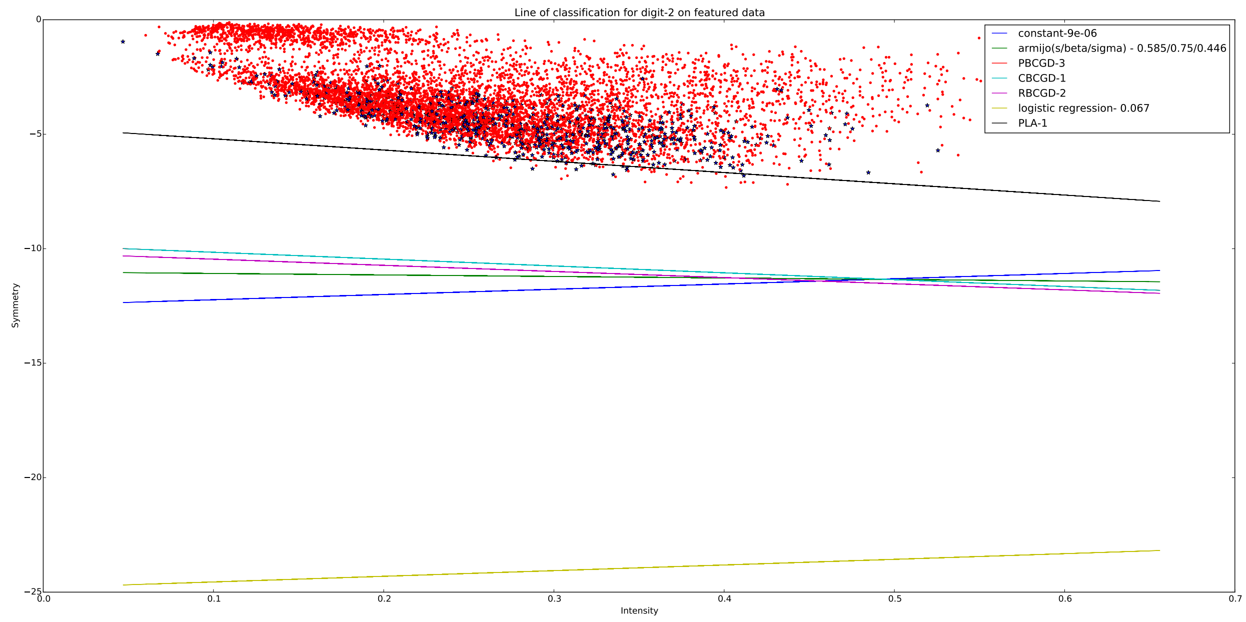

Figure 10: Classifiers of different models for digit 2 on raw data

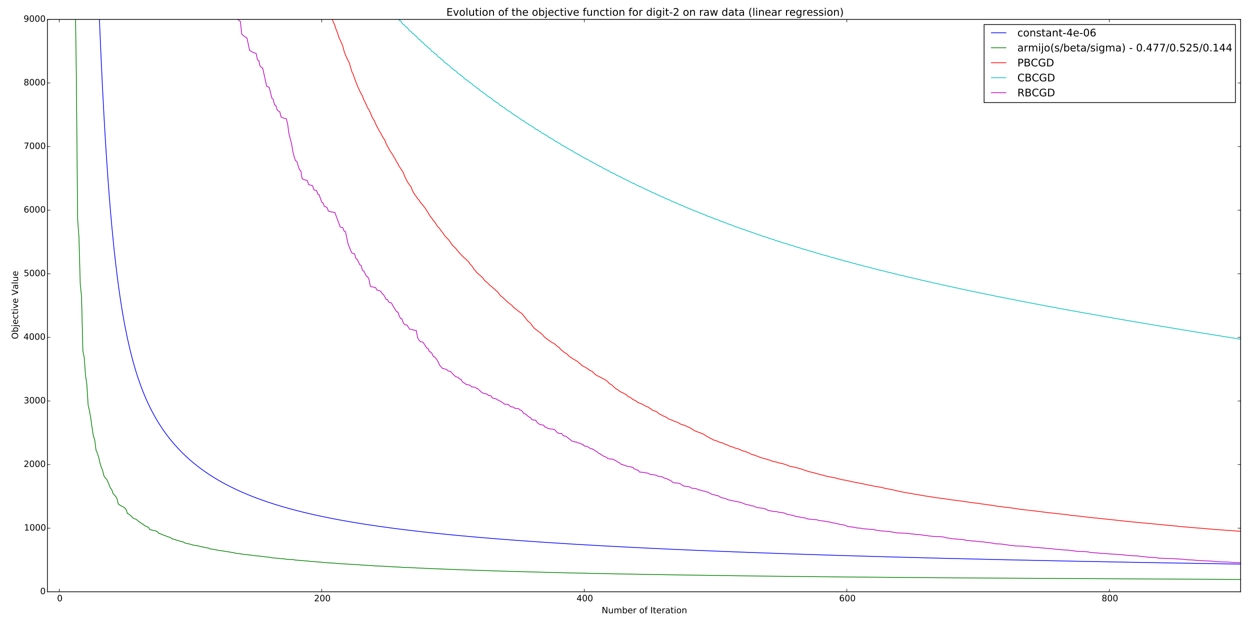

Figure 11: Objective value reduction of linear regression for digit 2 on raw data

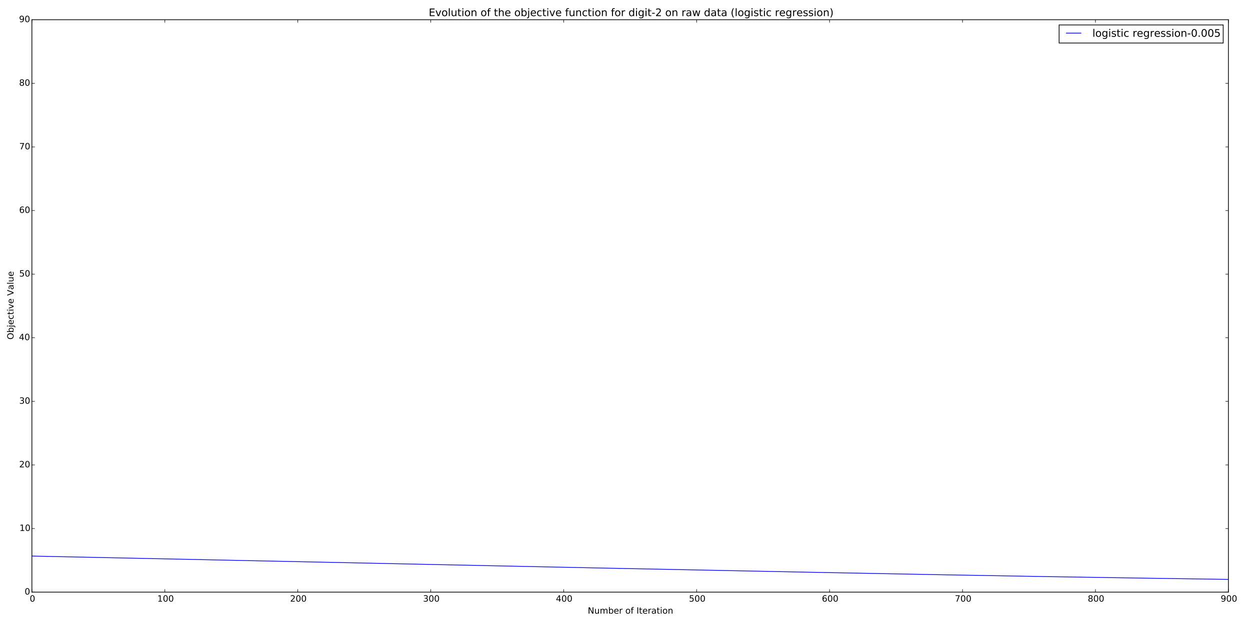

Figure 12: Objective value reduction of logistic regression for digit 2 on raw data

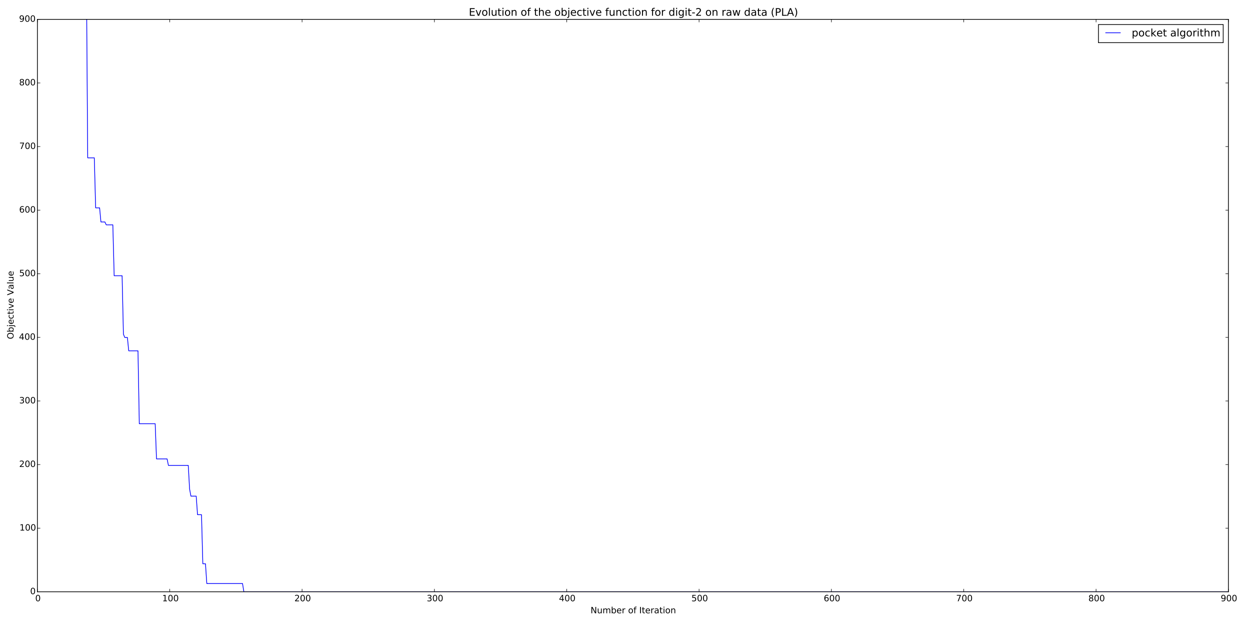

Figure 13: Objective value reduction of PLA model for digit 2 on raw data