: Apply stochastic subgradient algorithm to classify hand writing digit (by SVM)

Model: Let $\mathbf{a} \in \mathbb{R}^d$ be a vector representing, for example, an image, an audio track, or a fragment of text. Our goal is to design a classifier, i.e. a function that associates to each vector $\mathbf{a}$ a positive or negative label based on a desired criterion, for example the fact that the image contains or not a cat, that the audio track contains or not English speech, or that the text is or not a scientific paper.

The vector $\mathbf{a}$ is classified by looking at the sign of a *linear scoring function* $\langle \mathbf{a}, \mathbf{x} \rangle$. The goal of learning is to estimate the parameter $\mathbf{x} \in \mathbb{R}^d$ in such a way that the score is positive if the vector $\mathbf{a}$ belongs to the positive class and negative otherwise. In fact, in the standard SVM formulation the goal is to have a score of *at least* 1 in the first case, and of at most -1 in the second one, imposing a *margin*.

The parameter $\mathbf{x}$ is estimated or *learned* by fitting the scoring function to a training set of n example pairs $(\mathbf{a}_i, b_i), i = 1, \ldots, n$. Here $b_i \in \{-1, 1\}$ are the *ground truth labels* of the corresponding example vectors. The fir quality is measured by a loss function which, in standard SVM, is the *hinge loss*:

$$L_i(\mathbf{x}; \mathbf{a}_i, b_i) = \max\{0, 1 - b_i(\mathbf{a}_i^T \mathbf{x})\}.$$

Note that the hinge loss is zero only if the score $\langle \mathbf{a}_i, \mathbf{x} \rangle$ is at least 1 or at most -1, depending on the label b_i.

Fitting the training data is usually insufficient. In order for the scoring function *generalize to future data* as well, it is usually preferable to trade off the fitting accuracy with the *regularity* of the learned scoring function $\langle \mathbf{a}_i \mathbf{x} \rangle$. Regularity in the standard formulation is measured by the norm of the parameter vector $\|\mathbf{x}\|^2$. Averaging the loss on all training samples and adding to it the regularizer weighed by a parameter γ yields the *regularized loss objective*:

$$L(\mathbf{x}) = \frac{1}{N} \sum_i^N \max\{0, 1 - b_i(\mathbf{a}_i^T \mathbf{x})\} + \gamma \mathbf{x}^T \mathbf{x}.$$

Note that this objective function is *convex*, so that there exists a single global optimum.

In the given dataset, we have two features, 'intensity' and 'symmetry', which gives decision variable $\tilde{x} = [x_1, x_2]$. However, the linear prediction model should be in the following form:

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

That is, for any given data point $\mathbf{a}$, the model produces the prediction of the above form. So here x_0 is the variable that characterizes the 'intercept' of the line. Therefore, we need the following classifier:

$$x_1 a_1 + x_2 a_2 + x_0 = 0,$$

which precisely defines a line on a 2-dimensional plane. Therefore, we need another decision variable, so the final variable vector is $\mathbf{x} = [x_1, x_2, x_0]^T \in \mathbb{R}^3$.

Given the task of classifying a digit and the rest, for all instances i correspond to the designated letter, the label $b_i = +1$, and for the rest of the instances, the label $b_j = -1$.

Algorithm: *Stochastic Gradient Descent* (SGD) minimizes directly the primal SVM objective:

$$L(\mathbf{x}) = \frac{1}{N} \sum_i^N L_i(\mathbf{x}; \mathbf{a}_i, b_i) + \gamma \mathbf{x}^T \mathbf{x}.$$

SGD performs gradient steps by considering at each iteration one term $L_i(\mathbf{x}; \mathbf{a}_i, b_i)$ selected at random from this average. In its most basic form, the algorithm is:

- Start with $\mathbf{x}_0 = 0$.
- For $r = 1, 2, \ldots, T$:
  - Sample one index i in $1, \ldots, n$ uniformly at random.
  - Compute a subgradient g^r of $L_i(\mathbf{x}; \mathbf{a}_i, b_i)$ at $\mathbf{x}^r$.
  - Compute the learning rate α^r.
  - Update $x^{r+1} = x^r - \alpha^r g^r$.

Provided that the learning rate α^r is chosen correctly, this simple algorithm is guaranteed to converge to the minimizer $\mathbf{x}^*$ of L.

Result: Two tasks were performed and the results are shown in the figures below. The tasks are separating digit '0' and the rest, '1' and the rest, the results are presented in Fig. 1 to 6, and 7 to 12, respectively. The performance in terms of error rates on both training and testing data is shown in Fig. 1 and 7 for two the two tasks. Note that each of the algorithms was performed for entire 100 iterations (or passes) and the elapsed time was recorded along with the error rates at every 10 iterations. From the figures, generally, the logistic regression model has the worst performance on both training and testing. The PLA method leads to the result where the training error and testing error are closed, whereas some other methods have better training error rates that have a large gap between testing error rates. For example, a linear regression model with a constant stepsize or Armijo rule based gradient method can reach a point below 20% training error but more than 85% testing error rate. Among all the implemented methods, the linear regression model with a randomized block-2 coordinate gradient method in both tasks has a satisfactory performance because of both the low training and testing errors. The linear SVM with stochastic subgradient methods did not perform particularly well. Fig. 2 and 8 show the classification lines drawn by the algorithms.

In Fig. 3 and 9, objection value reduction is compared by the same stochastic subgradient method but with different stepsize values at different iteration. With $\alpha^r = \frac{c}{\sqrt{r}}$ being a larger step size, the objective value decreased faster than $\alpha^r = \frac{c}{r}$. And since it is not guaranteed that the objective value can be reduced at each iteration like the gradient descend method, in some iterations, the objective value increase, but generally, it drops toward the optimum point. Fig. 4 and 10 records the objective value reduction in different tasks with PLA models. Since the pocket algorithm only accept a solution when the objective value can be decreased and the tasks are non linearly separable, the best objective value is stored and could stay the same for a deal number of iterations. The performances of gradient descent and coordinate descent methods are compared as presented in Fig. 5 and 11, where the RBCGD can obtain a good solution in around 30 iterations. In both Fig. 6 and 12, although the objective value is decreasing, the performance on error rates is the worst as mentioned before.

Analysis: In Gradient Descent (GD) optimization, we compute the cost gradient based on the complete training set; hence, we sometimes also call it *batch GD*. In case of very large datasets, using GD can be quite costly since we are only taking a single step for one pass over the training set; thus, the larger the training set, the slower our algorithm updates the weights and the longer it may take until it converges to the global cost minimum.

In SGD, we don't accumulate the weight updates as we've seen for GD. Instead, we update the weights after each training sample. Here, the term 'stochastic' comes from the fact that the gradient based on a single training sample is a 'stochastic approximation' of the 'true' cost gradient. Due to its stochastic nature, the path towards the global cost minimum is not 'direct' as in GD, but may go 'zig-zag' (like Fig. 3, for example). However, it has been shown that SGD almost surely converges to the global cost minimum if the cost function is convex.

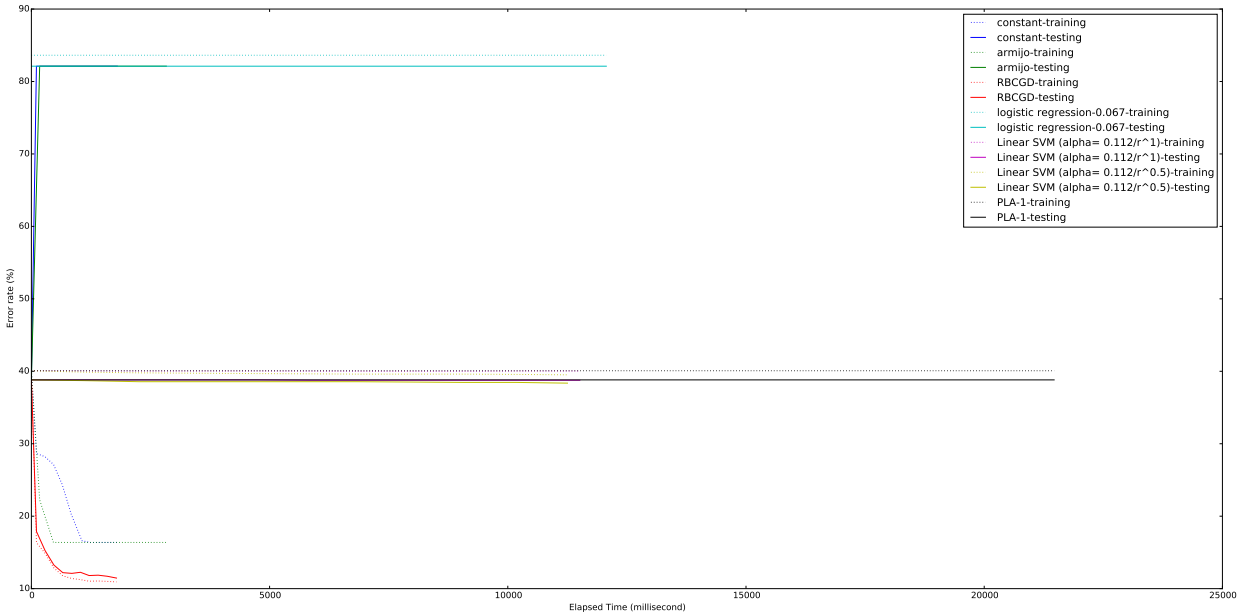

Figure 1: Error rate on classifying digit 0

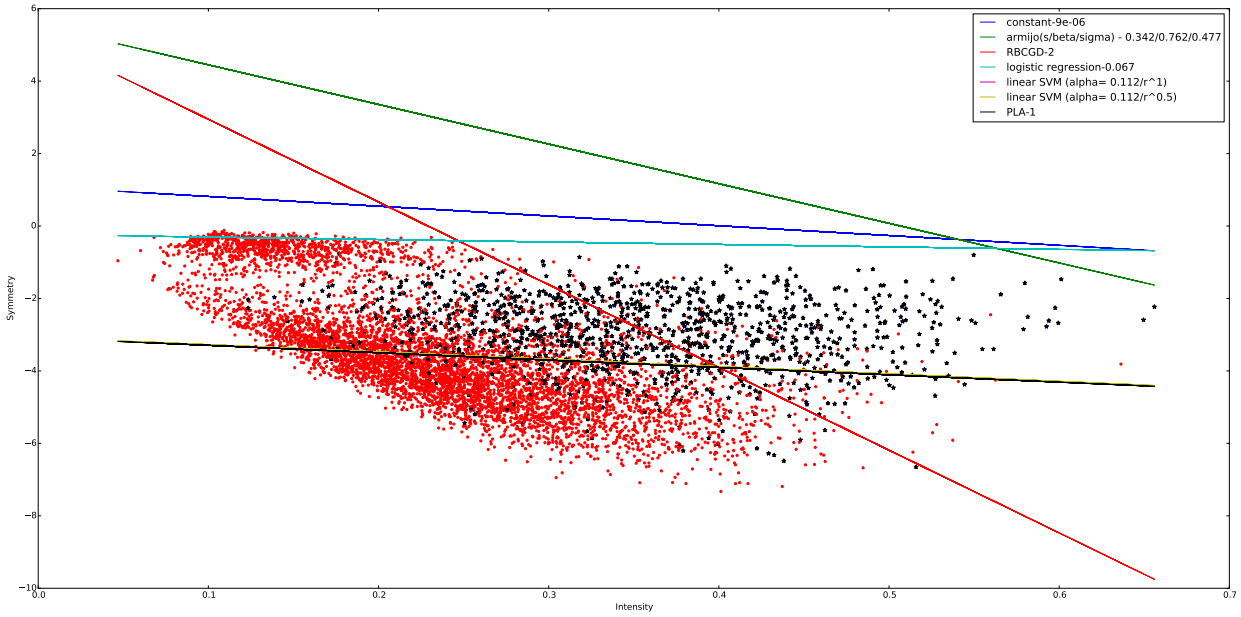

Figure 2: Line of classification for digit 0

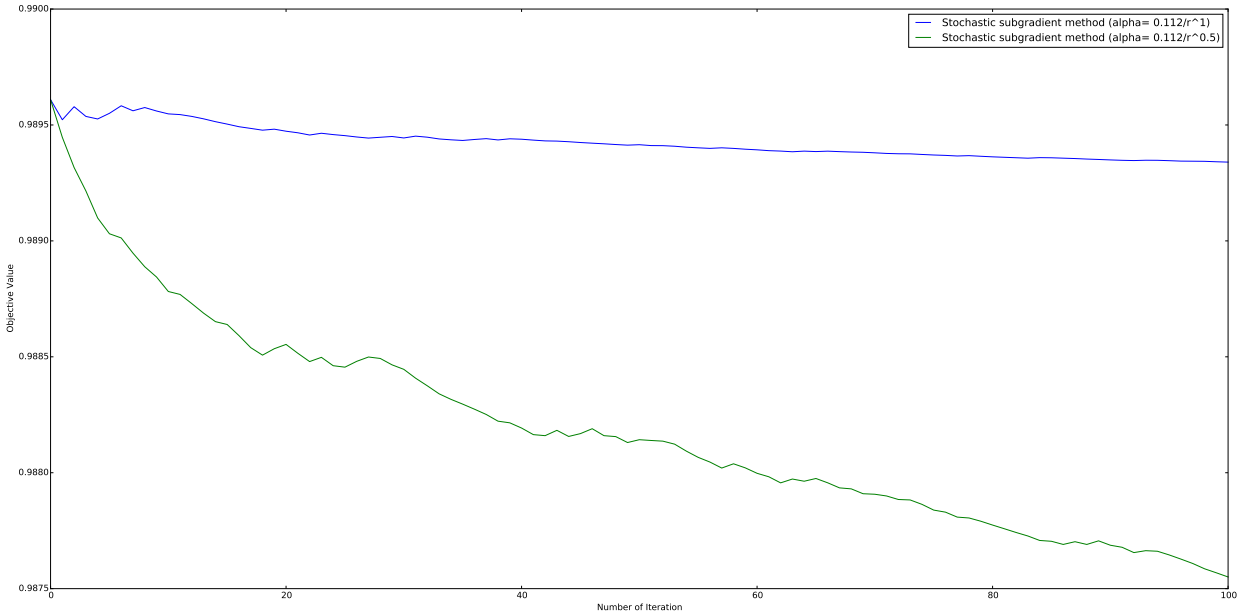

Figure 3: Objective value of linear SVM for digit 0

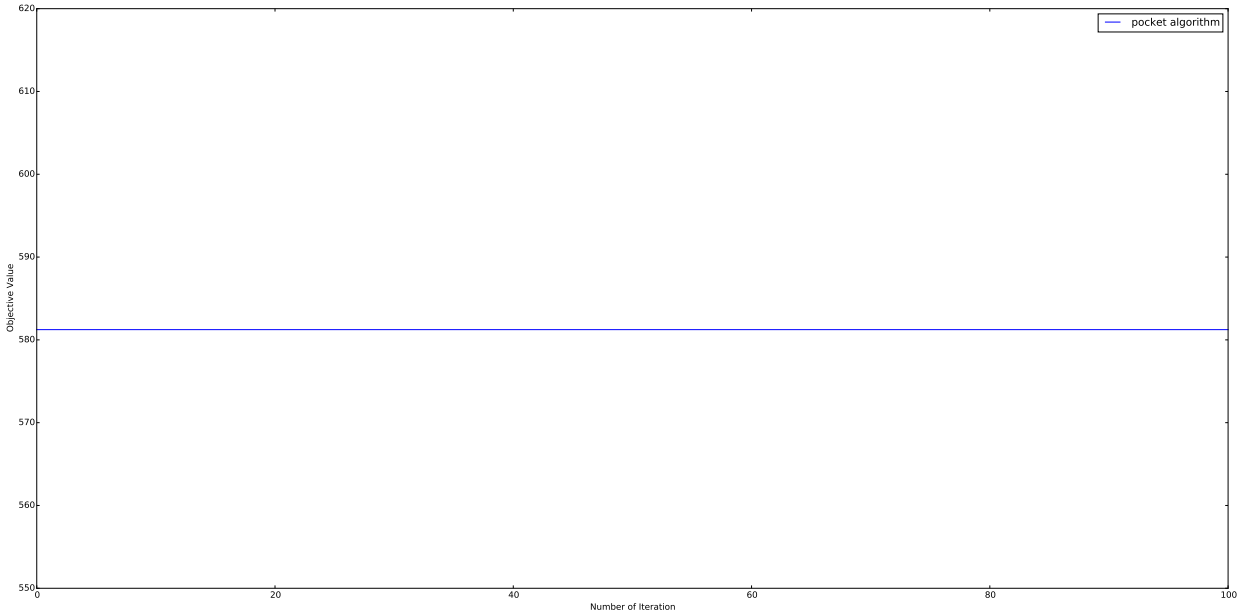

Figure 4: Objective value of PLA for digit 0

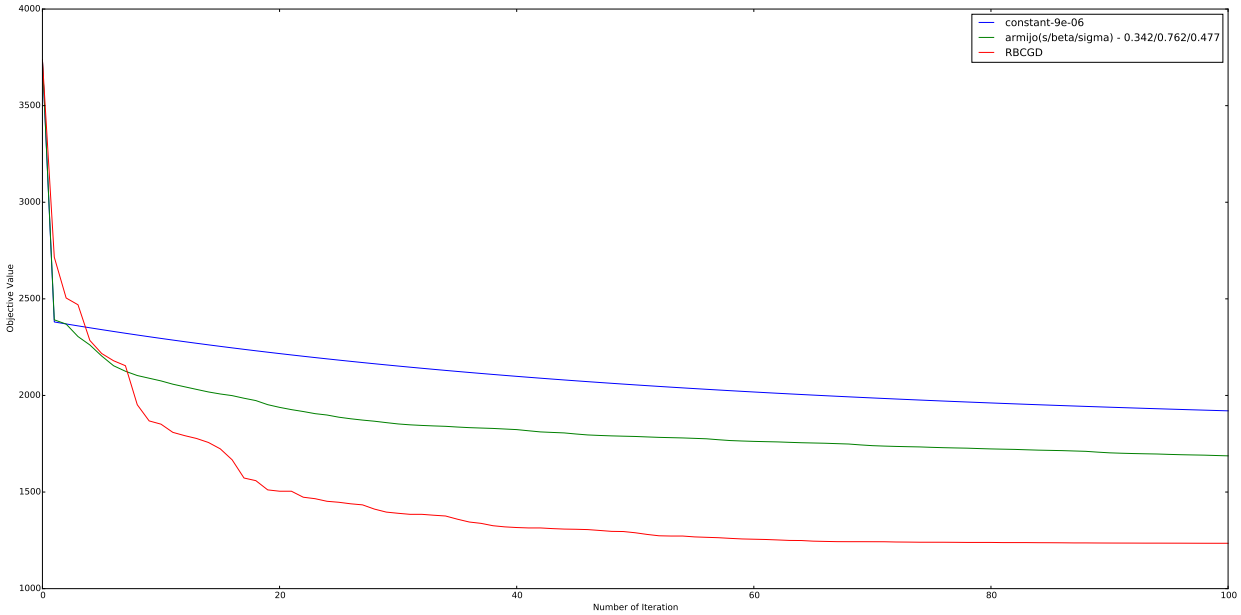

Figure 5: Objective value of linear regression for digit 0

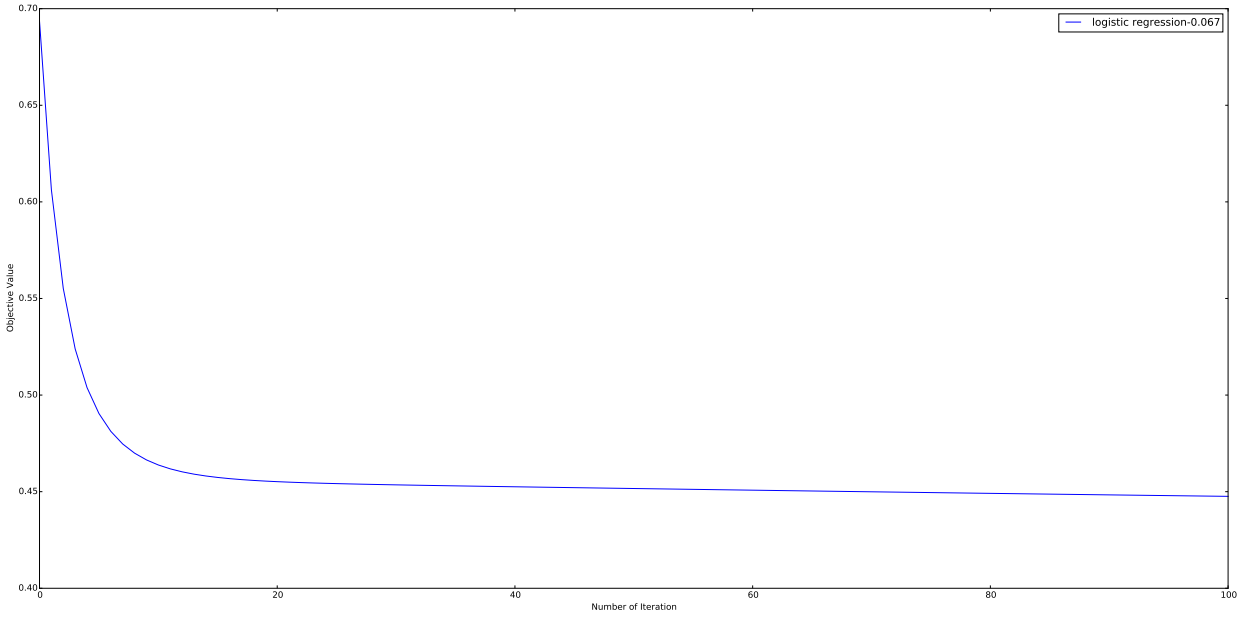

Figure 6: Objective value of logistic regression for digit 0

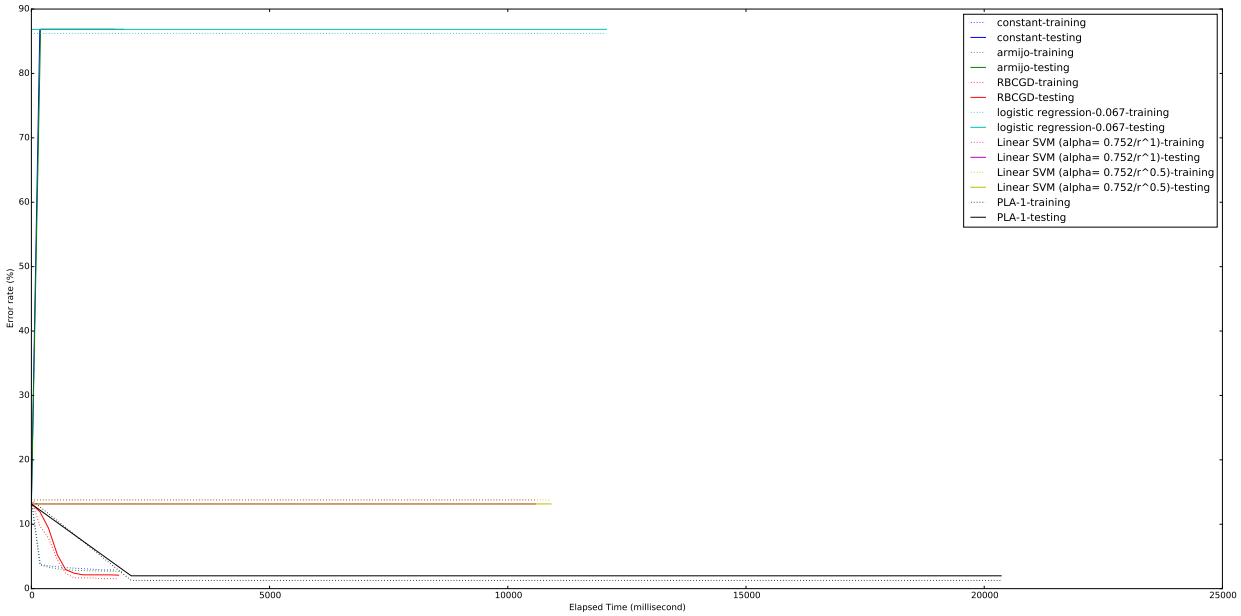

Figure 7: Error rate on classifying digit 1

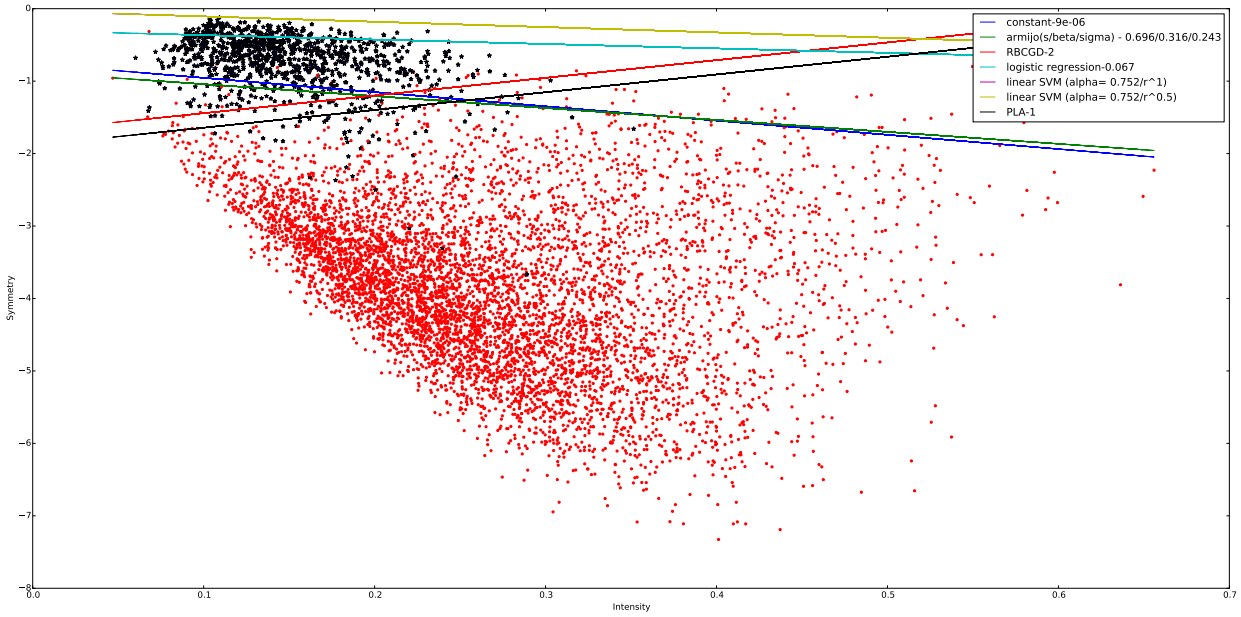

Figure 8: Line of classification for digit 1

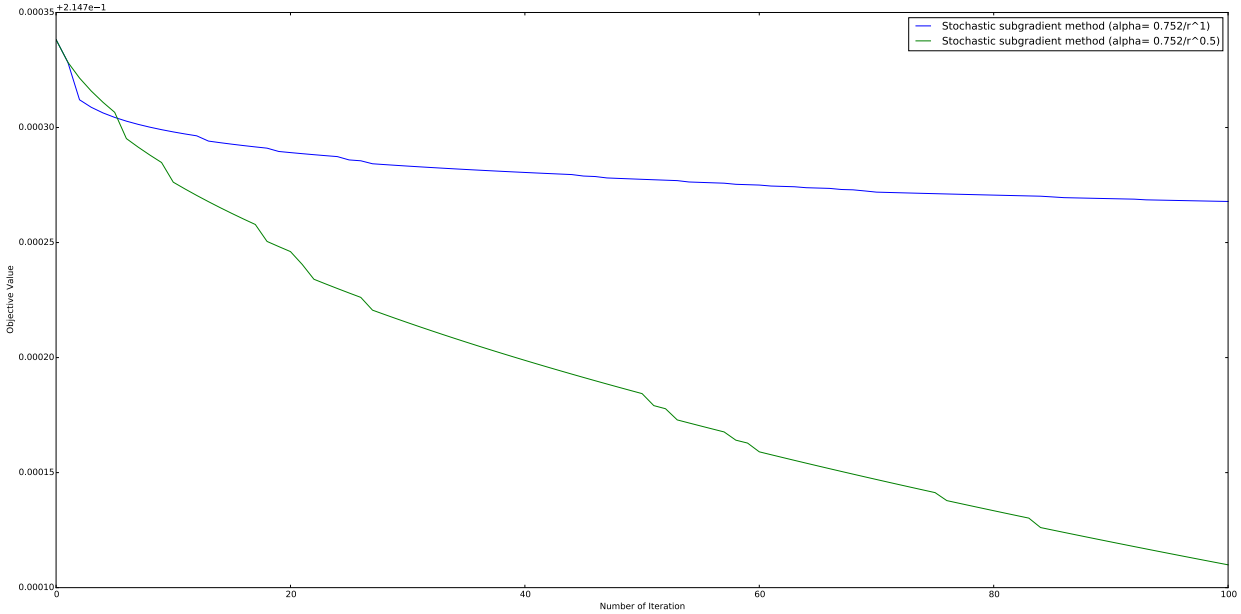

Figure 9: Objective value of linear SVM for digit 1

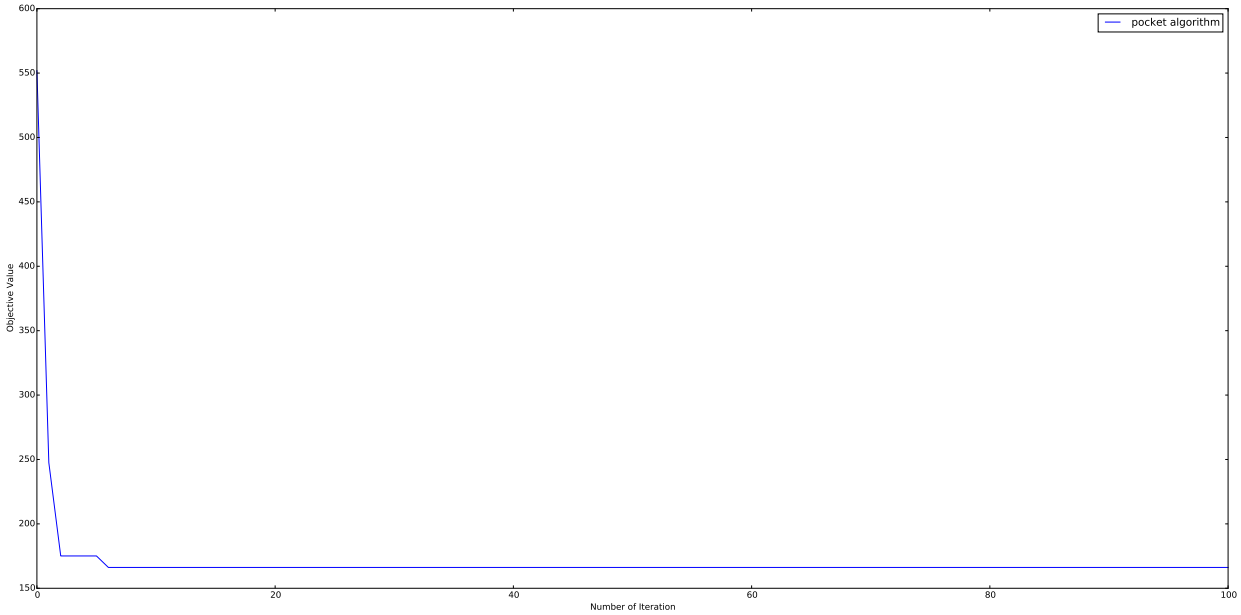

Figure 10: Objective value of PLA for digit 1

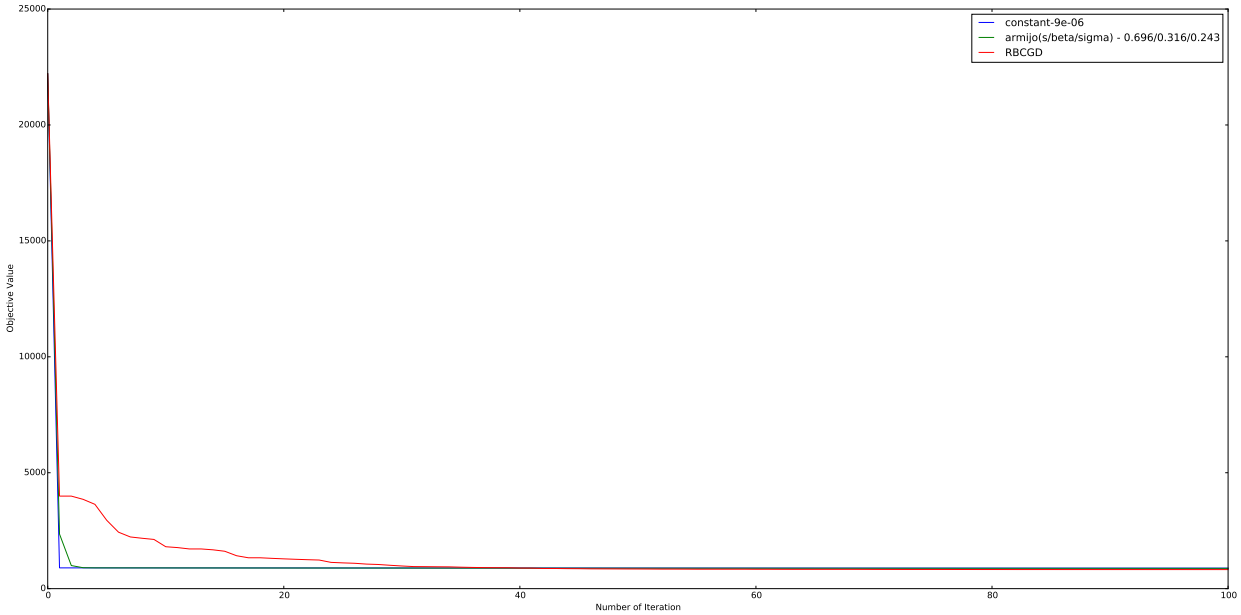

Figure 11: Objective value of linear regression for digit 1

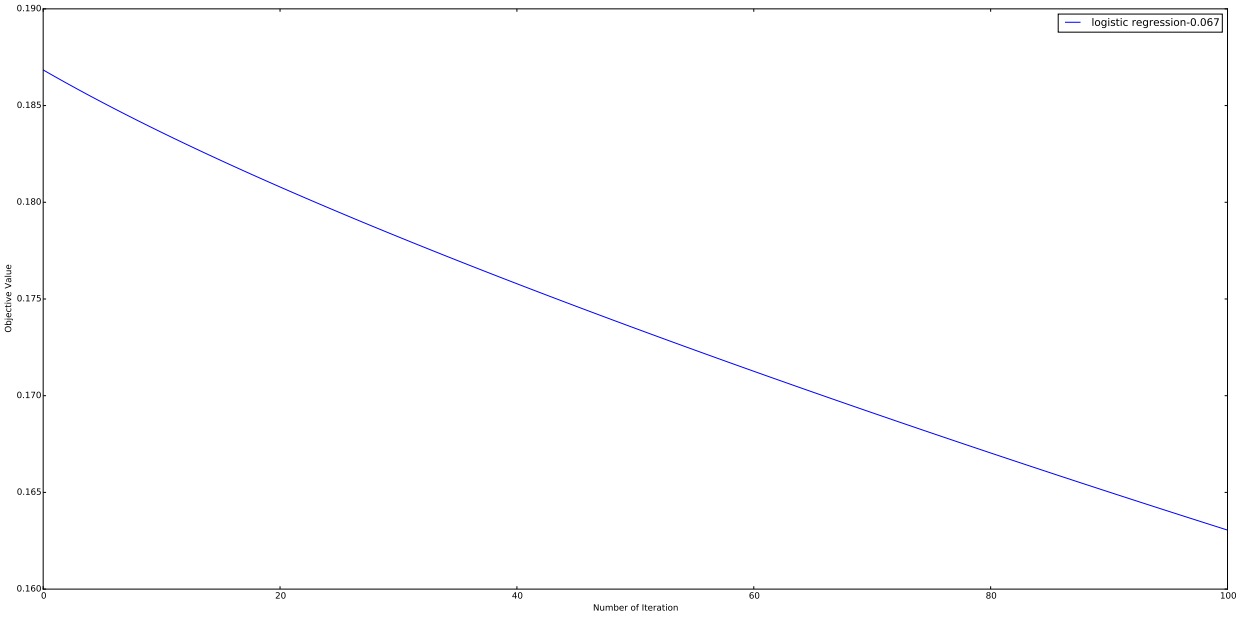

Figure 12: Objective value of logistic regression for digit 1