Stat 548 HW3

Jiahao Yang

May 18, 2018

Problem 0 Certify you read the HW Policies

0.1 List of Collaborators

I discussed problem 4 with Liqun Li.

0.2 List of Acknowledgements

Problem 3, 3.1: Matrix Inverse Lemma, T. Lienart, https: /www.stats.ox.ac.uk$\tilde{\text{l}}$ienartblog_linalg_invlemmas.html

0.3 Certify that you have read the instructions

I have read and understood these policies.

Problem 1 Gaussian Random Projections and Inner Products

Proof: Applying norm preservation theorem to the vector $u+v$ and $u-v$, with probability at least $1-4e^{-(\epsilon^2-\epsilon^3)m/4}$, we have

$$(1-\epsilon)\parallel u+v\parallel^2 \leq \parallel \phi(u+v)\parallel^2 \leq (1+\epsilon)\parallel u+v\parallel^2,$$

$$(1-\epsilon)\parallel u-v\parallel^2 \leq \parallel \phi(u-v)\parallel^2 \leq (1+\epsilon)\parallel u-v\parallel^2 .$$

Since we know that

$$\begin{aligned} 4\phi(u)\cdot\phi(v) &= \parallel \phi(u+v)\parallel^2 - \parallel \phi(u-v)\parallel^2 \\ &\geq (1-\epsilon)\parallel u+v\parallel^2 - (1+\epsilon)\parallel u-v\parallel^2 \\ &= 4u\cdot v - 2\epsilon(\parallel u\parallel^2 + \parallel v\parallel^2) \\ &\geq 4u\cdot v - 4\epsilon. \end{aligned}$$

Thus, we have $u\cdot v - \phi(u)\cdot\phi(v) \leq \epsilon$. The proof of the other direction is similar. And we finally have

$$Pr(|u\cdot v - \phi(u)\cdot\phi(v)| \geq \epsilon) \leq 4e^{-(\epsilon^2-\epsilon^3)m/4}$$

.

Problem 2 LSH for Angle Similarity

2.1

Vector p_1 and p_2 always define a plane and the angle between them is measured in this plane. The following figure is a "top-view" of the plane containing p_1 and p_2.

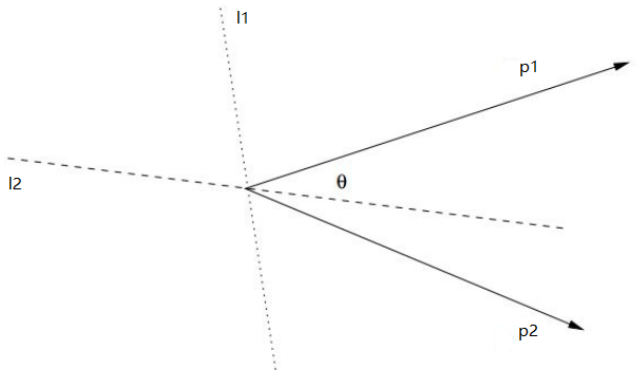

Figure 1: Two vectors make an angle θ

Suppose we pick a hyperplane through the origin. This hyperplane intersects the plane of p_1 and p_2 in a line. Figure 1 suggests two possible hyperplanes, one whose intersection is the dashed line l_2 and the other's intersection is the dotted line l_1. To pick a random hyperplane, we actually pick the normal vector to the hyperplane, say u. The hyperplane is then the set of points whose dot product with u is 0.

First, consider a vector u that is normal to the hyperplane whose projection is represented by l_2 in Figure 1; that is, p_1 and p_2 are on different sides of the hyperplane. Then the dot products $u \cdot p_1$ and $u \cdot p_2$ will have different signs.

On the other hand, the randomly chosen vector u could be normal to a hyperplane likel_1 in Figure 1, that is p_1 and p_2 are on the same side of the hyperplane. In that case, both $u \cdot p_1$ and $u \cdot p_2$ have the same sign.

Let $d(p, p') = angle(p, p')$. Then the probability that the randomly chosen vector is normal to a hyperplane that looks like the l_1 is $\frac{2\pi - 2d(p,p')}{2\pi} = 1 - \frac{d(p,p')}{\pi}$. That is

$$P(h(p) = h(p')) = P(sign(u \cdot p) = sign(u \cdot p')) = 1 - \frac{d(p, p')}{\pi}.$$

2.2

If $d(p, p') \leq \theta$,

$$P(h(p) = h(p')) \geq 1 - \frac{\theta}{\pi}.$$

If $d(p, p') \geq c\theta$,

$$P(h(p) = h(p')) \leq 1 - \frac{c\theta}{\pi}.$$

Thus, $P_1 = 1 - \frac{\theta}{\pi}$, $P_2 = 1 - \frac{c\theta}{\pi}$

2.3

By the LSH Theorem, if we choose $L = n^c$ when we use LSH algorithm, then with probability greater than $1 - 1/poly(n)$, the query time is $O(n^c)$, the space needed is $O(n^{1+c})$ and the

construction time is $O(n^{1+c})$ where $c = \frac{log(1/P_1)}{log(1/P_2)}$.

Problem 3 (Dual) Coordinate Ascent

3.1

Consider an invertible matrix $\mathbf{M}$ made of invertible blocks $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ with appropriate dimensions then the inverse of $\mathbf{M}$ can be expressed in terms of the inverses of the blocks with:

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{pmatrix} \quad and \quad \mathbf{M}^{-1} = \begin{pmatrix} \mathbf{W} & \mathbf{X} \\ \mathbf{Y} & \mathbf{Z} \end{pmatrix},$$

We know that $\mathbf{M}\mathbf{M}^{-1} = \mathbf{I}$. Then, we have

$$\begin{cases} \mathbf{AW}+\mathbf{BY} &= \mathbf{I} \\ \mathbf{AX}+\mathbf{BZ} &= \mathbf{0} \\ \mathbf{CW}+\mathbf{DY} &= \mathbf{0} \\ \mathbf{CX}+\mathbf{DZ} &= \mathbf{I} \end{cases}$$

Solving this system, we get

$$\begin{cases} \mathbf{W} &= (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} \\ \mathbf{Y} &= -\mathbf{D}^{-1}\mathbf{C}\mathbf{W} \\ \mathbf{Z} &= (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} \\ \mathbf{X} &= -\mathbf{A}^{-1}\mathbf{B}\mathbf{Z} \end{cases}$$

And we also know that $\mathbf{M}^{-1}\mathbf{M} = \mathbf{I}$. Similarly, we have

$$\begin{cases} \mathbf{Y} &= -\mathbf{Z}\mathbf{C}\mathbf{A}^{-1} \\ \mathbf{X} &= -\mathbf{W}\mathbf{B}\mathbf{D}^{-1} \end{cases}$$

Equating the expressions for $\mathbf{X}$ gives $\mathbf{A}^{-1}\mathbf{B}\mathbf{Z} = \mathbf{W}\mathbf{B}\mathbf{D}^{-1}$ from which we get

$$\mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1}.$$

Then, if we choose $\mathbf{A} = -\sqrt{\lambda}\mathbf{I}_d$, $\mathbf{B} = X^T$, $\mathbf{C} = X$, $\mathbf{D} = -\sqrt{\lambda}\mathbf{I}_n$, the equation will be

$$X^T(\lambda\mathbf{I} + XX^T)^{-1} = (\lambda\mathbf{I} + X^TX)^{-1}X^T$$

.

If $\lambda = 0$, the equality is no longer valid because XX^T may not be invertible.

3.2

We know that

$$G = \frac{1}{2}\sum_{j=1}^{n}\alpha_j^2 + \frac{1}{2\lambda}\sum_{(}j = 1)^n\sum_{k=1}^{n}\alpha_j\alpha_k x_j \cdot x_k - \sum_{j=1}^{n}\alpha_j y_j.$$

Take derivative with respect to α_i and set to 0, we have

$$\frac{\partial G}{\partial \alpha_i} = \alpha_i + \frac{(\sum_{j\neq i}\alpha_j x_j)\cdot x_i}{\lambda} + \alpha_i\frac{\parallel x_i \parallel^2}{\lambda} - y_i = 0$$

$$\alpha_i = \frac{y_i - \frac{1}{\lambda}(\sum_{j \neq i} \alpha_j x_j) \cdot x_i}{1 + \frac{\|x_i\|^2}{\lambda}}$$

3.3

The computational complexity of this update is $O(n + d)$.

3.4

The computational complexity of one stochastic descent update is $O(d)$.

3.5

The update rule in equation (3) can be written as

$$\alpha_i^{(i+1)} = \frac{y_i - \frac{1}{\lambda}(\sum_{j \neq i} \alpha_j x_j) \cdot x_i}{1 + \frac{\|x_i\|^2}{\lambda}} = \frac{y_i - w^{(i)} \cdot x_i - \alpha_i}{1 + \frac{\|x_i\|^2}{\lambda}} + \alpha_i^{(i)} = \alpha^{(i)} + \Delta\alpha_i^{(i)}$$

where $w^{(i)} = \frac{1}{\lambda} X^T \alpha^{(i)} = w^{(i-1)} + \frac{1}{\lambda} x_i \cdot \Delta\alpha_i^{(i-1)}$. Thus, the update rule in equation (3) is the same as the update rule of α_i in equation (5).

3.6

The computational complexity of this update is $O(d)$. Comparing to the original algorithm which has complexity $O(n + d)$, this new algorithm is more efficient.

Problem 4 Project milestone

4.1 Experimental design

In this experiment, I choose to use linear-SVM classifier and to use Heavy Ball algorithm to train this model with regularization parameter $\lambda = 500$, learning rate $\gamma = 1 \times 10^{-6}$ and mini-batch size $b = 16$.

SVM is a supervised machine learning algorithm which can be used for classification or regression problems. Although in this experiment, I choose to use linear SVM, SVM can uses a technique called the kernel trick to transform the data and then based on these transformations it finds an optimal boundary between the possible outputs. This model is flexible to deal with complex problem and it is convenient for me to modify my model to a non-linear model.

Because of the unbalanced dataset, I subsampled my negative training samples and retrain my model by hard negative mining. More details are shown in part 4.2 and part 4.5.

Instead of linear-SVM, I also tried linear regression model with logistic loss and logistic regression.

4.2 Negative examples

To deal with the unbalanced dataset, I subsampled my negative train examples and negative cross-validation examples.

For the original model before hare negative mining, after I get the ground truth labels for each bounding box, suppose I have n positive examples, I sample my 2n negative examples randomly

and construct my 3n training dataset and cross-validation set. For test set, I use all of the examples I have.

When I retrain my model using hard negative mining, I first choose 3n negative examples randomly and predict their labels and choose n negative examples which are assigned to be positive with highest probability and then I choose n negative examples randomly again to construct my 2n negative examples.

Also, I have tried to choose the first n negative examples from the negative examples which are assigned to be positive with highest probability in previous training set and then randomly sample n negative examples to construct my 2n negative examples.

Both of these two methods are reasonable because when we choose the negative examples which are assigned to be positive with highest probability, we actually choose those points which our model does not learn from them a lot. And we retrain our model on these points so that more information in these points can be learnt by our model. And we choose n new negative examples so that our model can learnt the negative points' information form new points in case overfitting our model.

4.3 Optimization plots

The optimization plots of 18 categories are shown below.

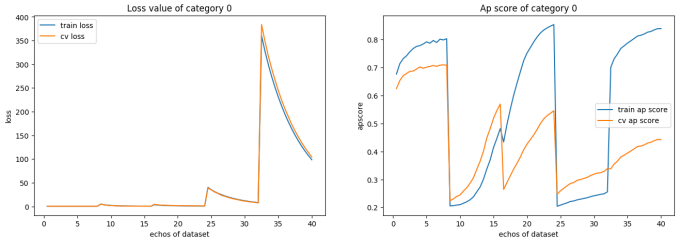

Figure 2: Loss and Ap score of category 0

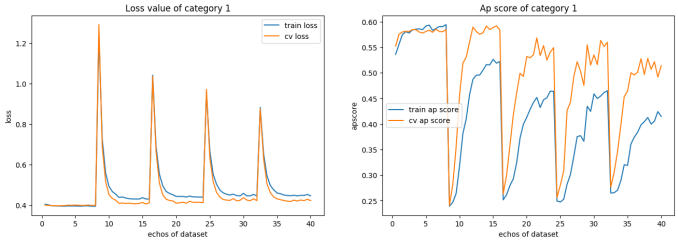

Figure 3: Loss and Ap score of category 1

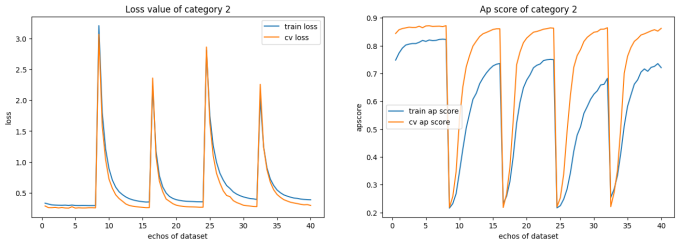

Figure 4: Loss and Ap score of category 2

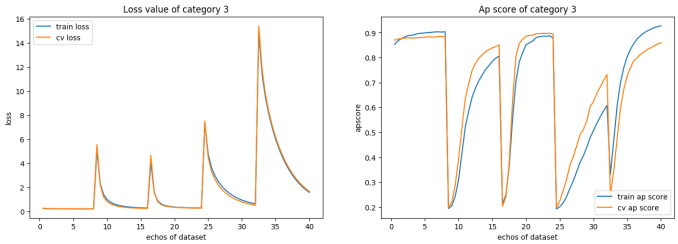

Figure 5: Loss and Ap score of category 3

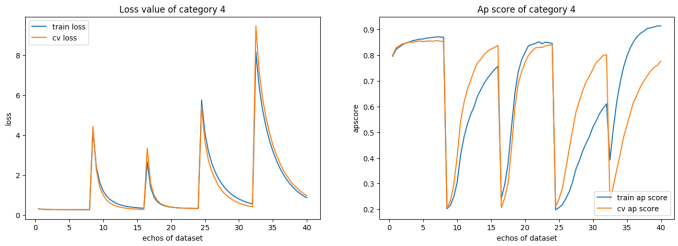

Figure 6: Loss and Ap score of category 4

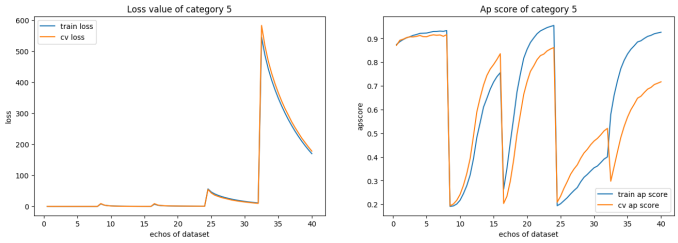

Figure 7: Loss and Ap score of category 5

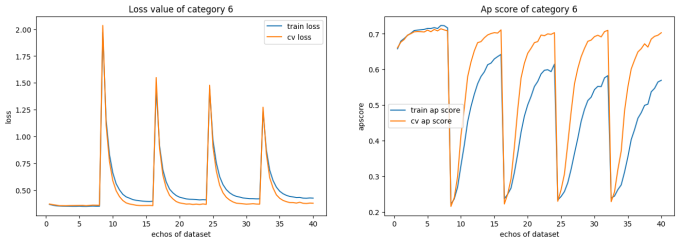

Figure 8: Loss and Ap score of category 6

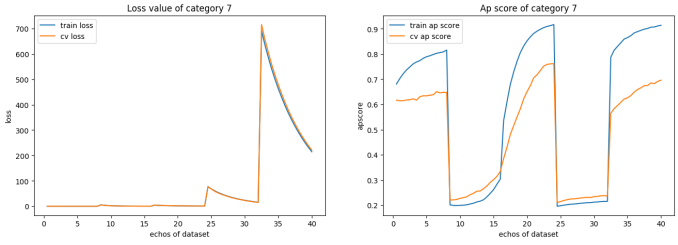

Figure 9: Loss and Ap score of category 7

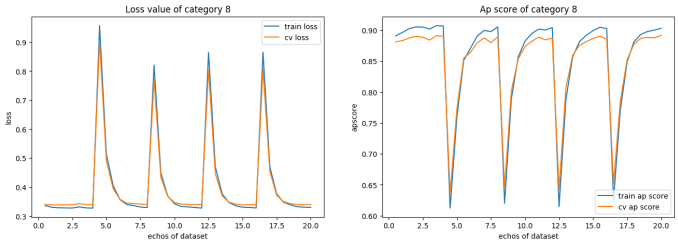

Figure 10: Loss and Ap score of category 8

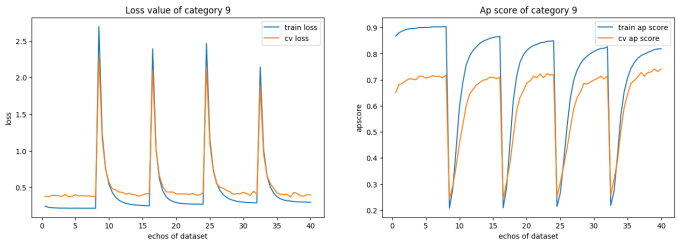

Figure 11: Loss and Ap score of category 9

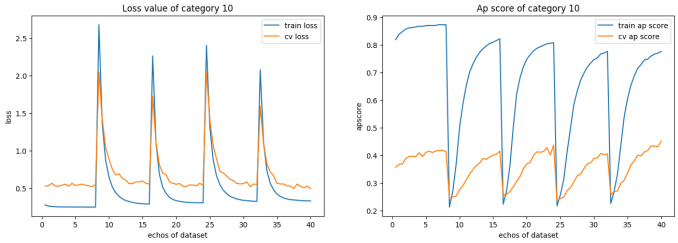

Figure 12: Loss and Ap score of category 10

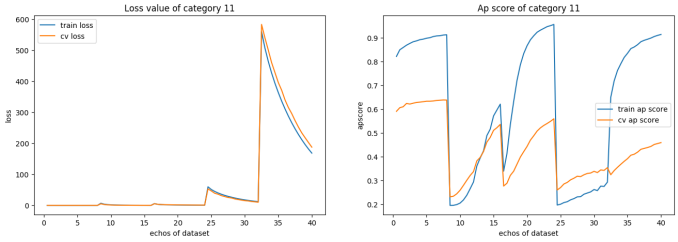

Figure 13: Loss and Ap score of category 11

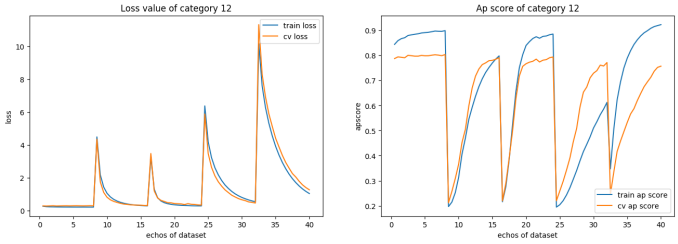

Figure 14: Loss and Ap score of category 12

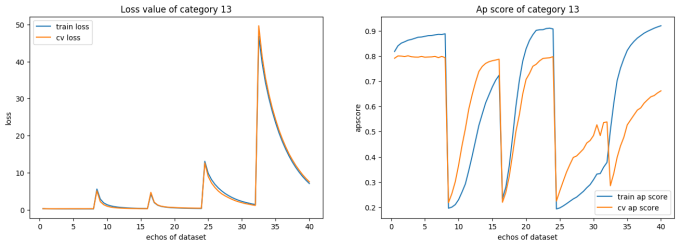

Figure 15: Loss and Ap score of category 13

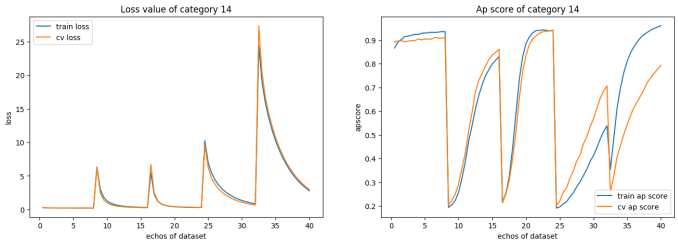

Figure 16: Loss and Ap score of category 14

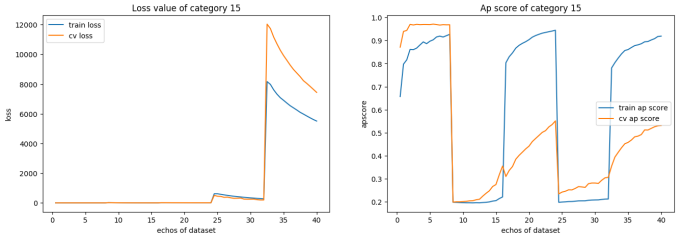

Figure 17: Loss and Ap score of category 15

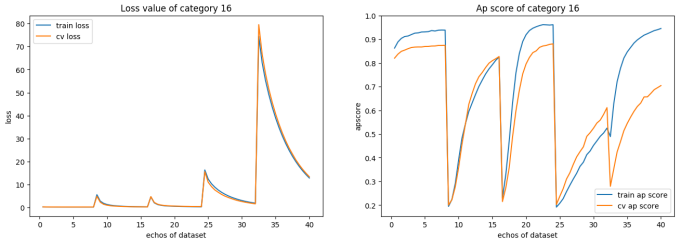

Figure 18: Loss and Ap score of category 16

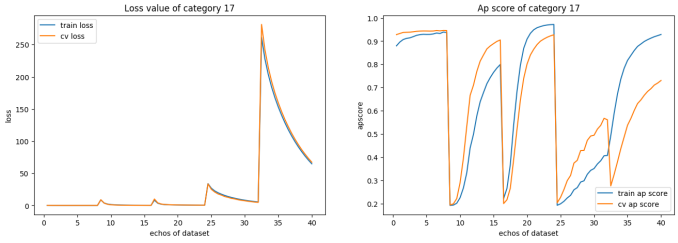

Figure 19: Loss and Ap score of category 17

4.4 Performance metrics

The performance matrix are show below.

Table 1: Ap score of 18 categories on test set(hard negative mining)

category number	0	1	2	3	4	5	6	7	8
C0	0.0603	0.0005	3.96E-05	0.0043	0.0048	0.0126	0.0065	0.0259	0.0031
C1	0.0100	0.0004	3.65E-05	0.0038	0.0015	0.0026	0.0091	0.0134	0.0035
C2	0.0028	0.0004	4.64E-05	0.0056	0.0085	0.0068	0.0117	0.0241	0.0029
C3	0.0001	0.0004	3.46E-05	0.0017	0.0005	0.0001	0.0128	0.0079	0.0031
C4	0.0017	0.0002	3.99E-05	0.0062	0.0107	0.0018	0.0180	0.0182	0.0036
category number	9	10	11	12	13	14	15	16	17
C0	NA	0.0019	NA	NA	3.62E-05	0.0109	NA	0.0124	0.1527
C1	NA	0.0016	NA	NA	6.43E-05	0.0019	NA	0.0266	0.1092
C2	NA	0.0015	NA	NA	1.92E-05	0.0136	NA	0.0173	0.0967
C3	NA	0.0010	NA	NA	8.66E-05	0.0005	NA	0.0009	0.0123
C4	NA	0.0021	NA	NA	1.31E-05	0.0008	NA	0.0074	0.0315

In this table, each row represents test Ap scores on test set of 18 categories for each model in the process of hard negative mining. Since in the test set, I do not have data of 4 categories, values in 4 columns are NA.

Table 2: Mean Ap score on test set of models in hard negative mining

models	C0	C1	C2	C3	C4
mAP	0.0211	0.0131	0.0137	0.0030	0.0073

From these two tables, we can find that the model become worse and worse but actually, this is reasonable since we keep adding new points which our model does not learnt or have not learnt them well.

Theoretically, the performance on test set should be better and better since our model will learn more and more information and the results in these two table are normal because I just retrain my model 5 times and most of the negative examples in training set have not been seen by my model. If I retrain my model enough times, the results will be better and better. And this is also the part that I will do in my final project.

4.5 Other details

I use a little trick to improve my loss function of linear-SVM model. Instead of using hinge loss function, I use squared-hinge loss function which is

$$l(x, y) = \frac{\lambda}{2} \parallel w \parallel + \frac{1}{2n} \sum_{i=1}^{n} (max(0, 1 - y_i(w \cdot x_i)))^2.$$

The reason why I do this is that the although the hinge loss is convex, it is not smooth. To improve the performance of using gradient-descent-based method, such as Heavy Ball, I square the hinge loss such that the loss function becomes convex and smooth.