CS578/STAT590: Introduction to Machine Learning **Fall 2014**

Problem Set 3

Gen Nishida *Handed In: November 16, 2014*

1 Questions

1. Fitting an SVM classifier by hand (source: Machine Learning, a probabilistic perspective. K Murphy)

   (a) Write down a vector that is parallel to the optimal vector w.
   The decision boundary is perpendicular to the vector $\phi(x_2) - \phi(x_1)$.

   $$\phi(x_2) - \phi(x_1) = [1, 2, 2]^T - [1, 0, 0]^T = [0, 2, 2]^T$$

   Thus, the vector that is parallel to the optimal vector w is $[0, 1, 1]^T$.

   (b) What is the value of the margin that is achieved by this w?
   The maximum margin is the half of the distance between two points in the 3d feature space. Thus,

   $$\frac{\|\phi(x_2) - \phi(x_1)\|}{2} = \frac{\|[0, 2, 2]^T\|}{2} = \sqrt{2}$$

   (c) Solve for w, using the fact the margin is equal to $1/\|w\|$.
   Let $w = k[0, 1, 1]^T$. Then,

   $$\|w\| = k\|[0, 1, 1]^T\| = \sqrt{2}k$$

   Since the margin is $\sqrt{2}$,

   $$\sqrt{2} = \frac{1}{\|w\|} = \frac{1}{\sqrt{2}k}$$

   By solving this, we obtain $k = 1/2$. Thus, $w = [0, 1/2, 1/2]^T$.

   (d) Solve for w_0 using your value for w and Equations 1 to 3.
   By substituting w of Equations 2 and 3, we get

   $$\begin{cases} y_1(w^T\phi(x_1) + w_0) = -([0, 1/2, 1/2]^T \cdot [1, 0, 0]^T + w_0) = -w_0 \geq 1 \\ y_2(w^T\phi(x_2) + w_0) = ([0, 1/2, 1/2]^T \cdot [1, 2, 2]^T + w_0) = 2 + w_0 \geq 1 \end{cases}$$

   By solving this, we obtain

   $$-1 \leq w_0 \leq -1$$

   Thus, $w_0 = -1$.

(e) Write down the form of the discriminant function $f(x) = w_0 + w^T\phi(x)$ as an explicit function of x.

$$f(x) = w_0 + w^T\phi(x) = -1 + [0, 1/2, 1/2]^T \cdot [1, \sqrt{2}x, x^2]^T = \frac{1}{2}x^2 + \frac{1}{\sqrt{2}}x - 1$$

2. We define a concept space C that consists of the union of k disjoint intervals in a real line. A concept in C is represented therefore using $2k$ parameters: $a_1 \leq b_1 \leq a_2 \leq b_2 \leq \cdots \leq a_k \leq b_k$. An example (a real number) is classified as positive by such concept iff it lies in one of the intervals. Give the VC dimension of H (and prove its correctness).

The answer is $2k$.

Proof:

Let $VC(k)$ be the VC dimension for k disjoint intervals in a real line. I prove by indiction that $VC(k) = 2k$ in the following. When $k = 1$, two examples have four patterns in total, and all the cases can be correctly classified (Figure 1). Thus, $VC(1) = 2$, which satisifies the above hypothesis.

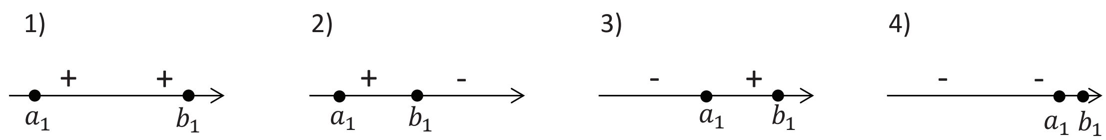

Figure 1: For the case of $k = 1$, two examples are correctly classified in all the four cases. Thus, $VC(1) = 2$.

Now, given that $VC(k-1) = 2(k-1)$, we want to show that two additional examples can be correctly classified by an additional interval. Assume without loss of generality that two additional numbers are greater than the existing $2(k-1)$ ones that are already classified. Then, there are only four cases in terms of the labels of two additional examples, and they are correctly classified by k-th interval in all cases (Figure 2).

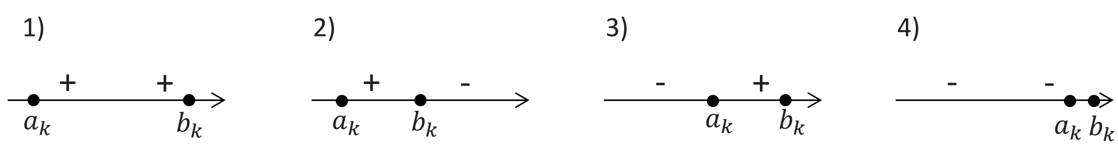

Figure 2: When adding k-th interval, two additional examples are correctly classified in all the four cases.

Thus,

$$VC(k) \geq VC(k-1) + 2 = 2(k-1) + 2 = 2k$$

Therefore, $VC(k)$ is at least $2k$ by induction. Also, Figure 3 shows a case in which $k+1$ positive examples and k negative examples cannot be correctly classified. This concludes $VC(k) = 2k$.

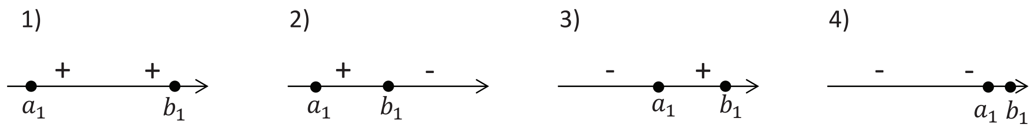

Figure 3: This figure shows a case in which $k+1$ positive examples and k negative examples cannot be correctly classified by k disjoint intervals. The right most positive example (red color) is classified incorrectly. Thus, $VC(k) < 2k+1$.

3. The Gradient Descent (GD) algorithm

   (a) Write in one sentence: what are the hyper parameters of the GD algorithm.

   The hyper parameters are the initial weight vector which you can randomly initialize if you want, learning rate that is the step size of updating the weight vector, a parameter λ that defines the impact of the regularizer, and the convergence criteria when to stop the iteration such as the maximum number of iterations.

   (b) Write in one sentence: What is the difference between $l1$ and $l2$ regularization.

   $l1$ regularization uses $l1$ norm of w as the regularization term, which encourages the sparsity, while $l2$ regularization uses $l2$ norm of w.

   (c) Write down the gradient descent algorithm applied to hinge loss with $l2$ regularization.

Algorithm 1 Gradient descent algorithm applied to hige loss with l2 regularization

procedure HingeRegularizedGD()
    $w = (0, \cdots, 0)$
    $b = 0$
    for $i = 0$ to $maxIterations$ **do**
        $\Delta w = (0, \cdots, 0)$
        $\Delta b = 0$
        for all training data (x_d, y_d) for $d = 1, \cdots, D$ **do**
            if $y_d(w \cdot x_d + b) \leq 1$ **then**
                $\Delta w = \Delta w + y_d x_d$
                $\Delta b = \Delta b + y_d$
        $\Delta w = \Delta w - \lambda w$
        $w = w + \eta \Delta w$
        $b = b + \eta \Delta b$
    $w = w / \|w\|$
    $b = b / \|w\|$
    return w and b

The objective function is

$$F(w) = \frac{\lambda}{2}\|w\|^2 + \sum_i \max\left(0, 1 - y_i(w^T x_i + b)\right)$$

Then, its partial derivative with regard to w and b are

$$\frac{\partial F(w)}{w} = \lambda w + \sum_i \begin{cases} -y_i x_i & (\text{if } y_i(w^T x_i + b) \leq 1) \\ 0 & \text{Otherwise} \end{cases}$$

and

$$\frac{\partial F(w)}{b} = \sum_i \begin{cases} -y_i & (\text{if } y_i(w^T x_i + b) \leq 1) \\ 0 & \text{Otherwise} \end{cases}$$

The algorithm is shown in Algorithm 1. Note that w and b are normalized in the end since our original objective was to maximize the margin while $\|w\|$=1 as follows:

$$w = \arg \max_{\|w\|=1} \left[\min_{(x,y)\in S} y(w^T x + b) \right]$$

2 Programming Assignment

For the programming assignment, I used SVM with the hinge loss function and $l1$ or $l2$ regularization as required. The pseudo code of gradient descent algorithm to solve this is shown in Algorithm 2, which is basically similar to Algorithm 1 except that both $l1$ and $l2$ are supported.

Feature Set

For the continuous values, I used all the intervals between two consequtive thresholds as attributes. For instance, if an original attribute a has three thresholds $1, 2, 3$, then the corresponding attributes will be $\{(a < 1), (a \geq 1; a < 2), (a \geq 2; a < 3), (a \geq 3)\}$. Also, I ignored the examples that contain missing values.

Hyper Parameters

There are five hyper parameters, $maxIterations$, $l1$ or $l2$, $stepSize$, $lambda$, and feature set. Unlike Perceptron algorithm, we want to minimize the objective function without worrying about overfitting because the regularization term somewhat avoids it. Thus, I used a very large number for maxIterations, say 20000, to get the gradient descent converged. For instance, Figure 4 shows the decrease of the objective function over the steps when $stepSize = 0.0001$ and $labmda = 0.01$, while Figure 5 shows some oscillation because of larger $stepSize$. Intuitively, wa do not want too much oscillation, but small $stepSize$ requires a large number of iterations to converge. The result of Figure 4 indicates that $maxIterations = 20000$ is enough to converge even for a very small $stepSize$.

For other hyperparameters, I employed a brute-force approach to find the best ones. For each combination of $regularization$ and $featureSet$, I tried all the combination of $stepSize \in \{0.0001, 0.001, 0.01, 0.1\}$ and $labmda \in \{0.001, 0.01, 0.1, 1\}$ in order to find the best ones that achieved the best accuracy on the validation data. One of the results are shown in Figure 6. Notice that the performance varies depending on $stepSize$ and $lambda$ although the

Algorithm 2 Gradient descent algorithm applied to hige loss with l1 and l2 regularization

procedure GD(maxIterations, regularization, η, λ)
    $w = (0, \cdots, 0)$
    $b = 0$
    for $iter = 0$ to $maxIterations$ **do**
        $\Delta w = (0, \cdots, 0)$
        $\Delta b = 0$
        for all training data (x_d, y_d) for $d = 1, \cdots, D$ **do**
            if $y_d(w \cdot x_d + b) \leq 1$ **then**
                $\Delta w = \Delta w + y_d x_d$
                $\Delta b = \Delta b + y_d$
        if regularization $==$ $l1$ **then**
            for $i = 0$ to N **do**
                if $w_i \geq 0$ **then**
                    $\Delta w_i = \Delta w_i - \lambda$
                else
                    $\Delta w_i = \Delta w_i + \lambda$
        else if regularization $==$ $l2$ **then**
            $\Delta w = \Delta w - \lambda w$
        $w = w + \eta \Delta w$
        $b = b + \eta \Delta b$
    $w = w / \|w\|$
    $b = b / \|w\|$
    return w and b

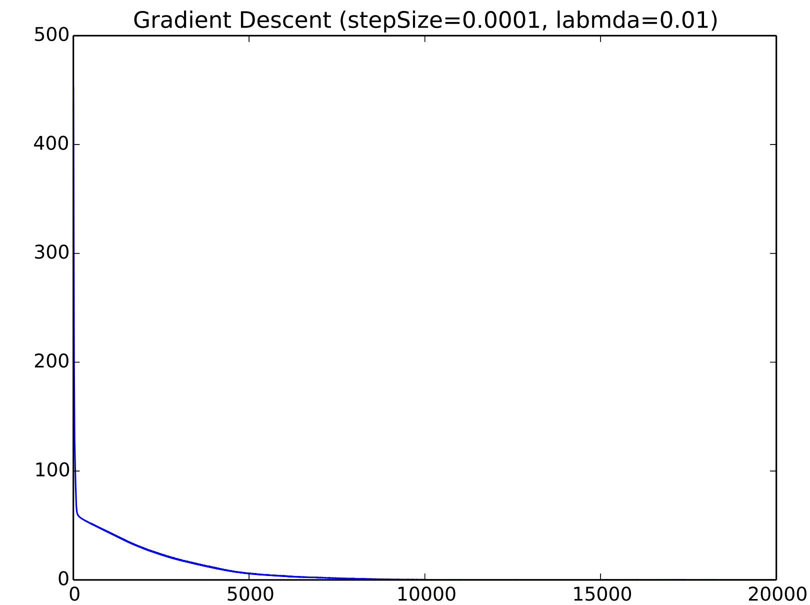

Figure 4: This figure shows the descrese of the objective over the steps when $stepSize = 0.0001$ and $labmda = 0.01$.

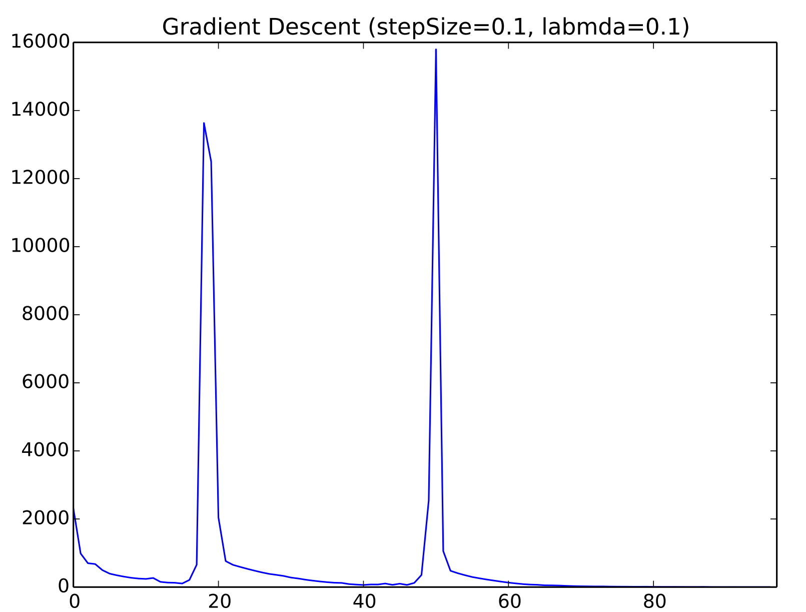

Figure 5: This figure shows the descrese of the objective over the steps when $stepSize = 0.1$ and $labmda = 0.1$. As $stepSize$ is relatively large, some oscillation occured.

difference is not significant. High $lambda$ resulted in relatively poor performance probably because it weighs less on the loss penalty, but we also want to have relatively large $lambda$ in order to avoid overfitting.

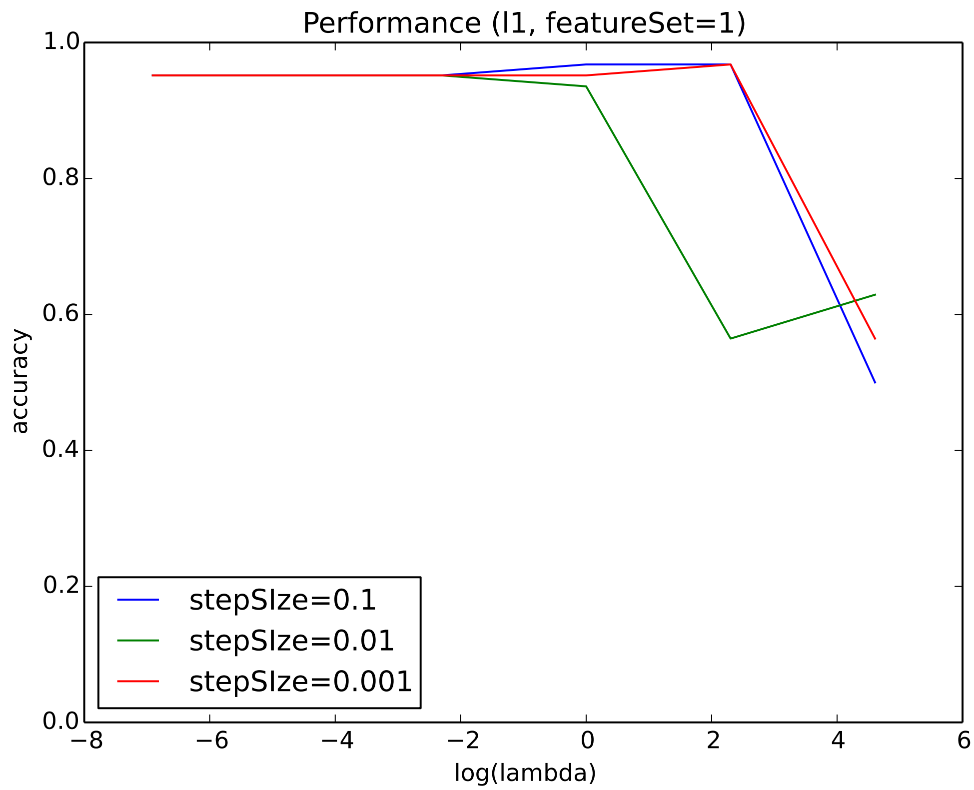

Figure 6: This figure shows the accuracy over the validation data using different pairs of $stepSize$ and $lambda$ when $regularization = l1$ and $featureSet = 1$. The performance drastically decreases when $lambda > 1$.

The best pairs of $stepSize$ and $labmda$ are listed in Table 1. For the following performance test over the test datset, I chose $lambda = 0.1$ because large $lambda$ avoids overfitting, and $stepSize = 0.01$ was used to reduce the number of iterations to converge.

Results

Using $stepSize = 0.01$ and $lambda = 0.1$, the classifier was learned and the performance was evaluated against the test data. The results are shown in Tables 2-6. Note that the performance of $featureSet = 2$ is very close to or exactly same as the one of $featureSet = 3$. This implies that the feature pairs have much larger weighs than the original attributes when $featuerSet = 3$ is used. Also, it is surprising that $featureSet = 1$ yields the best performance. The possible reason is that the feature pairs cause the overfitting.

featureSet	regularization	best stepSize	best lambda
1	l1	0.01	0.1
1	l2	0.1	0.1
2	l1	0.1	0.01
2	l2	0.01	0.1
3	l1	0.1	0.01
3	l2	0.1	0.01

Table 1: The best combination of *stepSize* and *lambda*

dataset	accuracy	precision	recall	F1
training data	1.0	1.0	1.0	1.0
validation data	0.951	0.928	0.962	0.945
test data	0.622	0.588	0.819	0.684

Table 2: The results of featurSet=1 with $l1$ regularization

Impact by the regularizer $l1$ and $l2$

In general, $l1$ regularization results in a sparse weight vector. To confirm this, I compare the resulting weight vectors between $l1$ and $l2$. The number of components whose absolute value is greater than 0.01 is 7 when $l1$ regularizer is used, while it is 11 when $l2$ regularizer is used. This result confirms that $l1$ regularizer leads to a sparse weight vector.

dataset	accuracy	precision	recall	F1
training data	1.0	1.0	1.0	1.0
validation data	0.951	0.928	0.962	0.945
test data	0.622	0.590	0.803	0.680

Table 3: The results of featurSet=1 with $l2$ regularization

dataset	accuracy	precision	recall	F1
training data	1.0	1.0	1.0	1.0
validation data	0.967	0.931	1.0	0.964
test data	0.573	0.549	0.819	0.657

Table 4: The results of featurSet=2 with $l1$ regularization

dataset	accuracy	precision	recall	F1
training data	1.0	1.0	1.0	1.0
validation data	0.956	0.928	0.962	0.945
test data	0.573	0.552	0.770	0.643

Table 5: The results of featurSet=2 with $l2$ regularization

dataset	accuracy	precision	recall	F1
training data	1.0	1.0	1.0	1.0
validation data	0.967	0.931	1.0	0.964
test data	0.573	0.549	0.819	0.657

Table 6: The results of featurSet=3 with $l1$ regularization

dataset	accuracy	precision	recall	F1
training data	1.0	1.0	1.0	1.0
validation data	0.967	0.931	1.0	0.964
test data	0.573	0.549	0.819	0.657

Table 7: The results of featurSet=3 with $l1$ regularization