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

Problem Set 3

Gen Nishida *Handed In: November 19, 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.

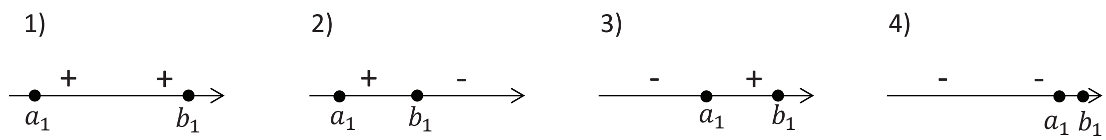

Figure 1: For the case of $k = 1$, two examples are correctly classified in all the four cases. Thus, $VC(1) = 2$.

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).

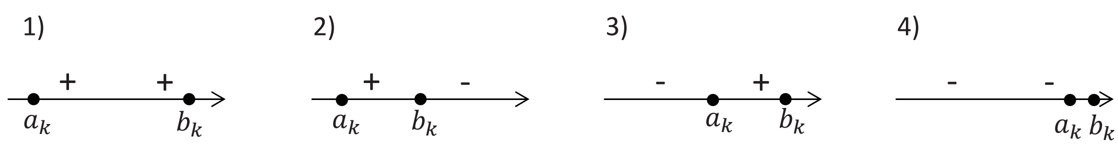

Figure 2: When adding k-th interval, two additional examples are correctly classified in all the four cases.

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. Thus, $VC(k) < 2k + 1$. This concludes $VC(k) = 2k$.

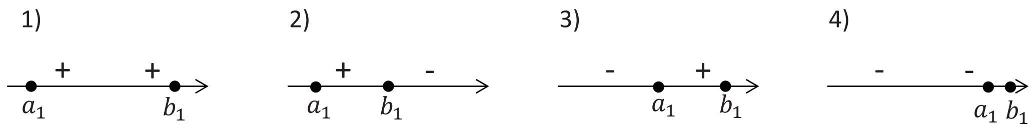

Figure 3: This figure shows a case in which $k+1$ positive examples and k negative examples cannot be correctly classified by k disjoint intervals. The right most positive example (red color) is classified incorrectly. Thus, $VC(k) < 2k+1$.

3. The Gradient Descent (GD) algorithm

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

   The hyper parameters are 1) how to define the initial parameters: initializing 0 or randomly, 2) the learning rate that is the step size of updating the parameters, and 3) the convergence criteria when to stop the iteration such as the maximum number of iterations and a threshold to judge whether it is coverged.

   (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 and leads to a simpler model, 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**
                $\Delta w = \Delta w + y_d x_d$
                $\Delta b = \Delta b + y_d$
        $\Delta w = \Delta w - \lambda w$
        $w = w + \eta \Delta w$
        $b = b + \eta \Delta b$
    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.

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.

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

```
procedure GD(maxIterations, regularization, η, λ)
    w = (0, ··· , 0)
    b = 0
    for iter = 0 to maxIterations do
        Δw = (0, ··· , 0)
        Δb = 0
        for all training data (x_d, y_d) for d = 1, ··· , D do
            if y_d(w · x_d + b) ≤ 1 then
                Δw = Δw + y_d x_d
                Δb = Δb + y_d
        if regularization == l1 then
            for i = 0 to N do
                if w_i ≥ 0 then
                    Δw_i = Δw_i − λ
                else
                    Δw_i = Δw_i + λ
        else if regularization == l2 then
            Δw = Δw − λw
        w = w + ηΔw
        b = b + ηΔb
        return w and b if they are converged
    return w and b
```

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 takes care of it. Thus, I used a very large number for maxIterations, say 20000, to get the gradient descent converged. Also, as it is shown in Algorithm 2, I checked the decrease in the objective function during the gradient descent and if it is within the threshold, the gradient descent stops. I used 0.0001 as threshold. Figure 4 ($featureSet = 1$, $stepSize = 0.001$, and $labmda = 0.4$) and 5 ($featureSet = 2$, $stepSize = 0.001$, and $labmda = 1.0$) show the decrease of the objective function over the iterations. A large $stepSize$ leads to more oscillation while a 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 this problem.

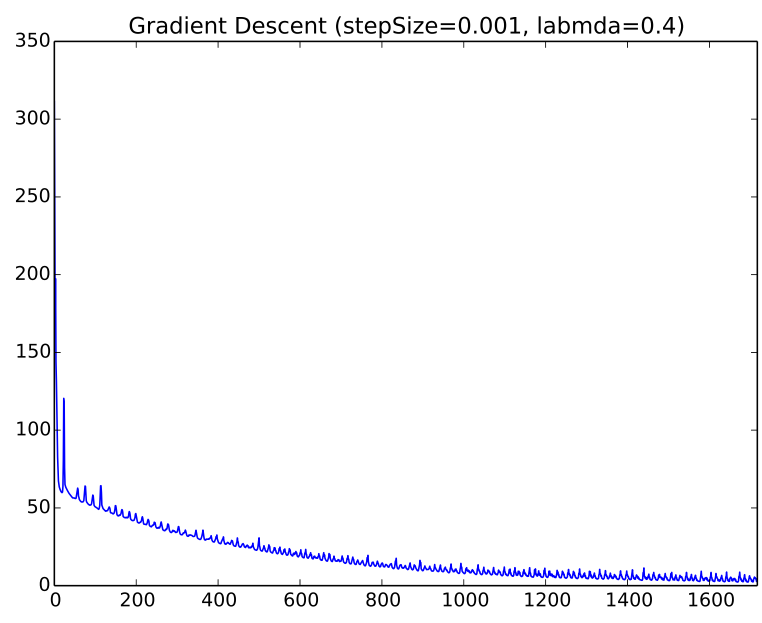

Figure 4: This figure shows the descrese of the objective over the iterations when $featureSet = 1$, $stepSize = 0.001$, and $labmda = 0.4$.

For $stepSize$ and $lambda$, I employed a brute-force approach to find the best values. For each combination of $regularization$ and $featureSet$, I tried all the combination of $stepSize \in \{0.001, 0.01, 0.1\}$ and $labmda \in \{0.05, 0.1, 0.2, 0.4, 1.0, 2.0\}$ in order to find the optimal values

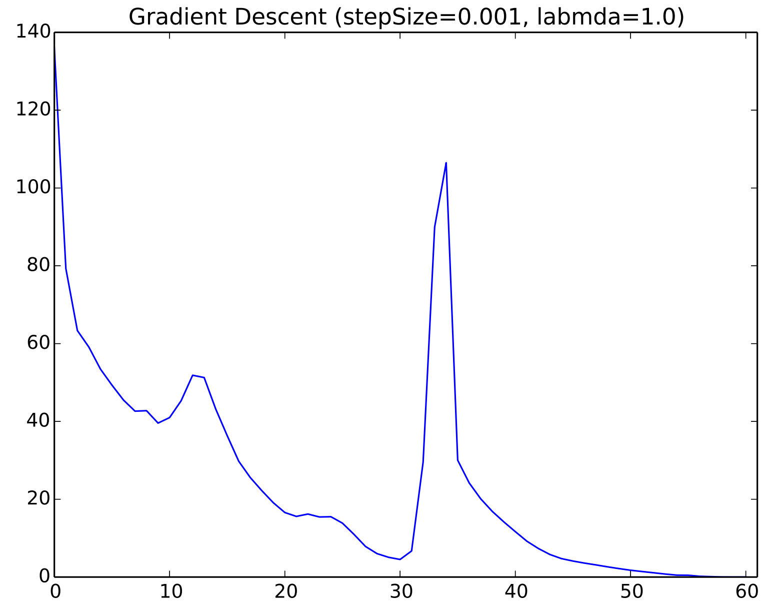

Figure 5: This figure shows the decrese in the objective over the iterations when $featureSet = 2$, $stepSize = 0.001$, and $lambda = 1.0$.

that achieve the best accuracy on the validation data. One of the results are shown in Figure 6. Notice that the performance significantly decreases when $lambda \geq 1$. This is resonable because the loss function becomes less important in the objective function. A large *lambda* avoids overfitting but may deteriorate the performance, while a small *lambda* can achieve better performance but may results in overfitting. Since we do not know the distribution of the test dataset beforehand, it is reasonable to use the performance on the validation data to choose values of *labmda* as well as other hyperparameters. Note that if the distribution of the test dataset is significantly different from the validation dataset, then the selected values of hyperparameters cannot necessarily yield a good performance on the test dataset.

The optimal values of *stepSize* and *labmda* that achieved the best performance on the validation data are listed in Table 1. For the following performance test, I used those values for hyperparameters.

featureSet	regularization	best stepSize	best lambda
1	l1	0.001	0.4
1	l2	0.001	0.4
2	l1	0.001	0.1
2	l2	0.001	0.2
3	l1	0.001	0.05
3	l2	0.001	0.1

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

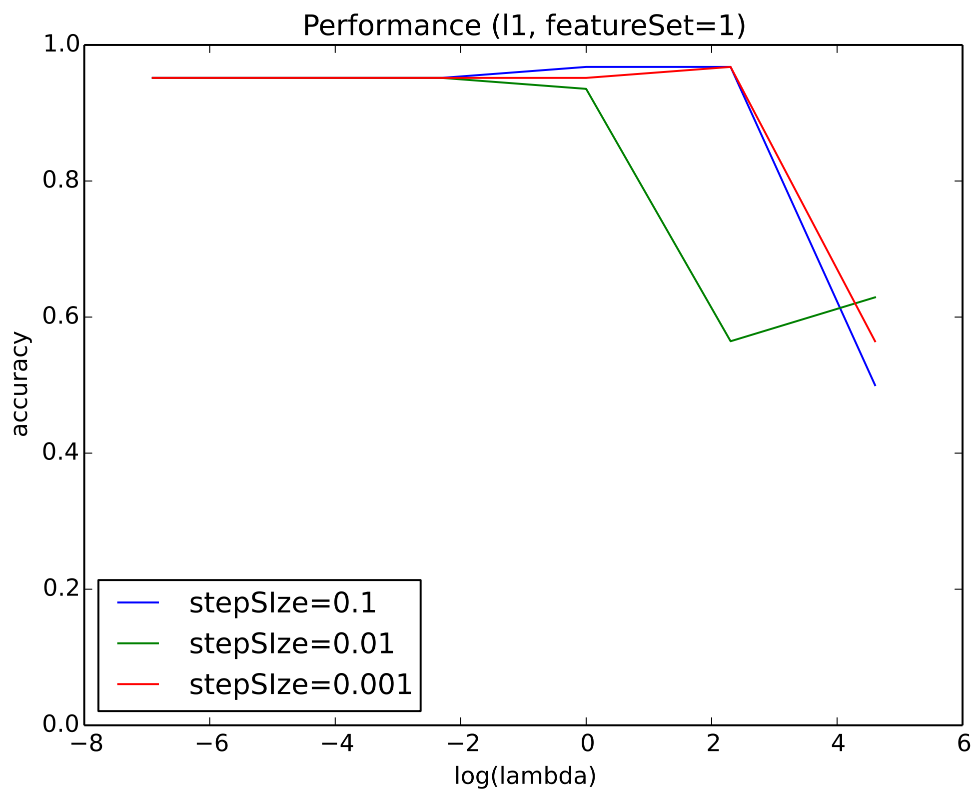

Figure 6: This figure shows the accuracy over the validation data using different pairs of $stepSize$ and $lambda$ when $regularization = l2$ and $featureSet = 1$. The performance drastically decreases when $lambda > 1$.

Results

Using the selected values of $stepSize$ and $lambda$ for each $featureSet$ and $regularization$, the classifier was learned and the performance was evaluated on the training, validation, and test dataset. The results are shown in Tables 2-7. Note that the performance of $featureSet = 2$ is very close to 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 even though the regularizer is included in the objective.

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	0.614	0.583	0.803	0.675

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

dataset	accuracy	precision	recall	F1
training data	0.989	0.990	0.985	0.987
validation data	0.967	0.931	1.0	0.964
test data	0.598	0.571	0.786	0.662

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	0.590	0.560	0.836	0.671

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	0.967	0.931	0.1.0	0.964
test data	0.606	0.573	0.836	0.680

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

Impact by the regularizer $l1$ and $l2$

In general, $l1$ regularizer results in a sparse weight vector compared to $l2$ regularizer. To confirm this, for $featureSet = 1$ I normalized the resulting weight vectors and compare them

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	0.606	0.573	0.836	0.680

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	0.606	0.573	0.836	0.680

Table 7: The results of featurSet=3 with $l2$ regularization

between $l1$ and $l2$. Out of 586 components of the weight vector, the number of components whose value is 0 is 313 when $l1$ regularizer is used, while it is 287 when $l2$ regularizer is used. This result confirms that $l1$ regularizer leads to a sparse weight vector. The sparsity is usually favorable because it leads to a simpler model, which indicates a more generalized model and requries less storage. Although it is known that $l1$ regularizer often leads to better performance in practice, there is usualy not a considerable difference between the two methods in terms of the accuracy of the resulting model [1, 2]. In fact, there is no significant difference in performace between $l1$ and $l2$ in the results.

References

[1] Gao, J., Andrew, G., Johnson, M., Toutanova, K. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of ACL*, pp. 824 - 831.

[2] Tsuruoka, Y., Tsujii, J., Ananiadou, S. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cummulative penality. In *Proceedings of ACL*, pp. 477 - 485.