Assignment1 SVM Report

A1655611 Xuanyu Zhao

1. Definitions for variables used in this report.

   $R(\alpha)$ — The expectation of the test error for a trained machine, representing generalization error.
   $R_{emp}(\alpha)$ — The "empirical risk", in fact the training error.
   $\{\mathbf{x}_i, y_i\}$ — training set. $i=1,\ldots\ldots,\ell$, $y_i \in \{-1, 1\}$, $\mathbf{x}_i \in R^d$.
   $\mathbf{w}$ — Weight vector of the separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$.
   b — Offset used to describe the separating hyperplane $\mathbf{w} \cdot \mathbf{x} + b = 0$.
   α_i — Lagrange multipliers α_i , $i = 1,\ldots\ldots,\ell$.
   L_P — Primal Lagrange function.
   L_D — Dual Lagrange function.
   ξ_i — positive slack variables, $i = 1,\ldots\ldots,\ell$.

2. Introduction to primal and dual problems.
   Support vector machine is an algorithm based on primal and dual problem transfer. Here I briefly introduce the primal and dual problems.
   Standard form of primal problem:
   minimize $f_0(\mathbf{x})$,
   s.t. $f_i(\mathbf{x}) \leqq 0$, $i = 1,\ldots\ldots, m;$
   $h_i(\mathbf{x}) = 0$, $i = 1,\ldots\ldots, p;$ $\mathbf{x} \in R^n$, optimal value is $P^*=\min f_0(\mathbf{x})$.
   We could form the Lagrange function for the primal problem:
   $$L(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{v}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} v_i h_i(\mathbf{x}),\ \boldsymbol{\lambda} \geqq \mathbf{0};$$
   then we define dual function $g(\boldsymbol{\lambda}, \mathbf{v})=\mathrm{Inf}\, L(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{v})$, which is the minimal of $L(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{v})$ for all $\mathbf{x} \in D$;
   The lower bound property holds: $g(\boldsymbol{\lambda}, \mathbf{v}) \leqq P^*$.

   Then we define the dual problem: find max $g(\boldsymbol{\lambda}, \mathbf{v})$, which means the best lower bound of P^*.
   Use d^* to denote the optimal value of dual problem, $d^* = \max g(\boldsymbol{\lambda}, \mathbf{v})$.

   The gap between d^* and P^* is called duality gap. There are 2 kinds of duality:
   Weak duality : $d^* \leqq P^*$, this always holds.
   Strong duality: $d^* = P^*$, usually holds for convex problems.

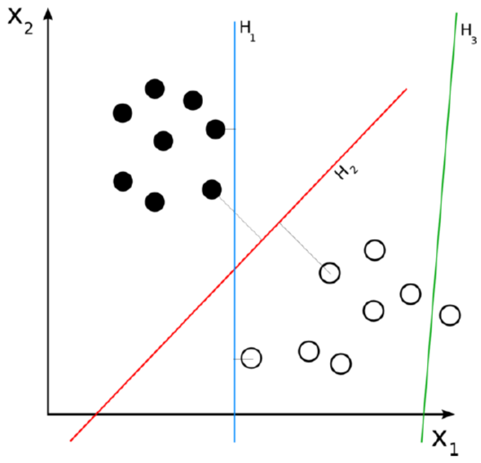
   Luckily that in SVM strong duality holds, therefore in SVM algorithm, we could solve the primal problem by solving the corresponding dual problem.

3. Description of SVMs
   Given some training data $\{ \mathbf{x}_i, y_i \}$, $i=1,\ldots\ldots,\ell$, $y_i \in \{-1, 1\}$, $\mathbf{x}_i \in R^d$. Suppose we have some separating hyperplane which separates the y=1points from the y=-1 points. The points $\mathbf{x}$ which lie on the hyperplane satisfy $\mathbf{w} \cdot \mathbf{x} + b = 0$, where w is normal to the hyperplane. Let d+ (d-) be the shortest distance from the separating hyperplane to the closest positive (negative) point. We define the "margin" of a separating hyperplane to be (d+) + (d-). SVM simply looks for the separating hyperplane with largest margin.

   3.1 Why max margin is good.
   In below graph, hyperplane H2 has the largest margin. SVM algorithm always aims to find a hyperplane with max margin. The main reason is to reduce the generalization error.
   If we just pick up any separating hyperplane which separates the training data well, it is not guaranteed to generalize well to new test data. However, if we choose the separating plane that

maximizes the margin, we will get much better generalization. Intuitively, by maximizing the margin we are squeezing out all the surplus capacity that came from using a high-dimensional feature space.

This can be justified by mathematics which shows that large margin separating hyperplanes have lower VC dimension and models with lower VC dimension have a smaller gap between the training and test error rates.



3.2 Primal and dual forms of hard margin SVMs

Hard margin SVMs enforce that all points are out of the margin. In other words, it requires the hyperplane perfectly separate all the positive points from all the negative points in training set. It doesn't permit any prediction error happen on training data.

Therefore, suppose that all the training data satisfy the following constraints:

$\mathbf{x_i} \cdot \mathbf{w} + b \geqq 1$, for $y_i = 1$;

$\mathbf{x_i} \cdot \mathbf{w} + b \leqq -1$, for $y_i = -1$;

These can be combined into:

$y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geqq 0$, for any i;

Geometrically the margin is the distance between two hyperplanes $\mathbf{x_i} \cdot \mathbf{w} + b = 1$ and $\mathbf{x_i} \cdot \mathbf{w} + b = -1$, which is $\frac{2}{||\mathbf{w}||}$ after calculation. We want to maximize $\frac{2}{||\mathbf{w}||}$, which is the same as minimize $||\mathbf{w}||$. For mathematical convenience, we substitute $||\mathbf{w}||$ with $\frac{1}{2}||\mathbf{w}||^2$.

So we deduce the primal form of hard margin SVMs:

minimize $\frac{1}{2}||\mathbf{w}||^2$,

subject to $y_i(\mathbf{x_i} \cdot \mathbf{w} + b) - 1 \geqq 0$;

We use Lagrange formulation to deduce the dual form, introducing positive Lagrange multipliers α_i, $i = 1, \ldots\ldots, \ell$. .

$$\text{Lp} = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{l} \alpha_i y_i(\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i,$$

We must now minimize Lp with respect to **w**, b, and simultaneously require that the partial derivatives of Lp with respect to all the α_i equal 0, all subject to the constraints $\alpha_i \geqq 0$. This is a convex quadratic programming problem, since the objective function is itself convex, and those points which satisfy the constraints also form a convex set.

Therefore we can equivalently solve the following "dual" problem: maximize Lp , subject to the constraints that the gradient of LP with respect to w and b equal 0, and subject also to the constraints that $\alpha_i \geqq 0$. Then we get

$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$, and $\sum_i \alpha_i y_i = 0$.

Then we substitute them into Lp to get the dual Lagrange formulation

L_D

$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$, from here we get the dual form of hard margin SVMs:

Maximize L_D,
Subject to $\alpha_i \geqq 0$, $\sum_i \alpha_i y_i = 0$.

3.3 Primal and dual forms of soft margin SVMs

Soft margin SVMs permit prediction error happen on training data. If there exists no hyperplane that can perfectly separate the positive and negative training points, the Soft Margin method will choose a hyperplane that splits the points as cleanly as possible, while still maximizing the distance to the nearest cleanly split points. The method introduces non-negative slack variables, ξ_i, which measure the degree of misclassification of the data $\mathbf{x}_i$.

$\mathbf{x}_i \cdot \mathbf{w} + b \geqq +1 - \xi_i$, for $y_i = 1$;
$\mathbf{x}_i \cdot \mathbf{w} + b \leqq -1 + \xi_i$, for $y_i = -1$;
$\xi_i \geqq 0, \forall i$

For an error to occur, the corresponding ξ_i must exceed unity, so $\sum_{i=1}^{l} \xi_i$ is an upper bound on the number of training errors. Hence a natural way to assign an extra cost for errors is to change the objective function to be minimized from $\frac{1}{2}||\mathbf{w}||^2$ to $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l} \xi_i$, where C is a parameter to be chosen by the user, a larger C corresponding to assigning a higher penalty to errors.

Therefore, the primal form changes to :

Minimize $\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l} \xi_i$,

subject to $y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i \geqq 0$, for any i, $\xi_i \geqq 0$;

Re-form the Lp and L_D, where the u_i are the Lagrange multipliers introduced to enforce positivity of the ξ_i.

$L_p = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \alpha_i\{y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 + \xi_i\} - \sum_{i=1}^{l} u_i \xi_i$,
$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$,

From here we get the dual form of soft margin SVMs:

Maximize L_D,
Subject to $0 \leqq \alpha_i \leqq C$, $\sum_i \alpha_i y_i = 0$.

The solution is again given by:
$\mathbf{w} = \sum_{i=1}^{Ns} \alpha_i y_i \mathbf{x}_i$, where *Ns* is the number of support vectors.

3.4 Concept of support vectors

So what are the support vectors? Recall the constraints $\sum_i \alpha_i y_i = 0$ in dual form. In the training data set, it turns out many $\alpha_i = 0$, and only a few α_i will be greater than 0. Those $\mathbf{x}_i$ with $\alpha_i > 0$ are called support vectors.

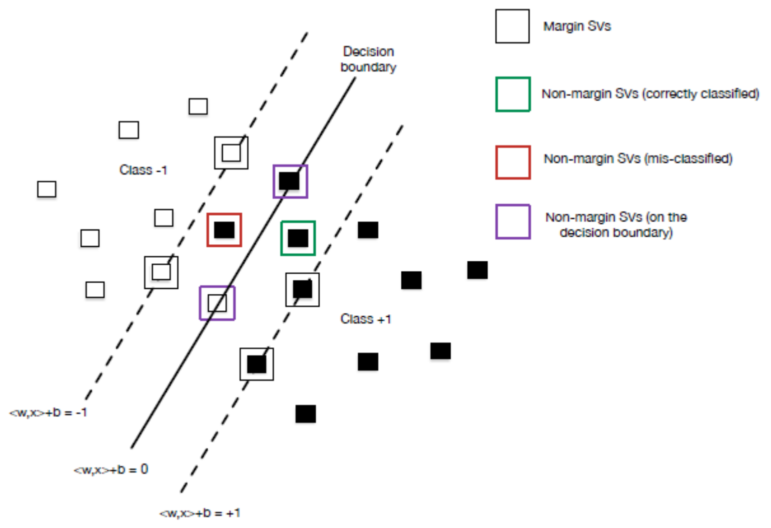
Therefore, support vectors are a subset of training data points. They lie closest to the decision surface (or hyperplane) and are the data points most difficult to classify. The decision function is fully specified by the support vectors.

In hard margin SVMs, support vectors lie on the margin. In soft margin SVMs, both margin and non-margin support vectors exist.

Two types of support vectors (shown in below diagram):

Margin SVs: $0 < \alpha_i < C$, $\xi_i = 0$, on the dash lines.

Non-margin SVs: $\alpha_i = C$, $\xi_i > 0$, thus violating the margin. More specifically, when $1 > \xi_i > 0$, correctly classified; when $\xi_i > 1$, it's mis-classified; when $\xi_i = 1$, on the decision boundary.


Margin SVs
Non-margin SVs (correctly classified)
Non-margin SVs (mis-classified)
Non-margin SVs (on the decision boundary)
Decision boundary
Class -1
Class +1
<w,x>+b = -1
<w,x>+b = 0
<w,x>+b = +1

4. Generalisation error and bounds

When using trained SVM classifier to classify new data points $\mathbf{x}^*$, the predicted value y^* is simply determined like below:

$y^* = \text{sign}\ (\mathbf{x}^* \cdot \mathbf{w} + b)$

Generalisation error is the expectation of the test error for a trained machine, also called true risk. Vapnik & Chervonenkis showed that generalisation error has an upper bound given by the empirical risk (training error) plus an additional term:

$R(\alpha) \leqq R_{emp}(\alpha) + \sqrt{\frac{h(log(\frac{2m}{h}+1)-log(\frac{\eta}{4})}{m}}$, where h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension. It is a measure of capacity or complexity of a set of functions.

We could understand this formula in this way:

Generalisation Error $\leqq$ Training Error + Complexity of Set of Models

If you take a high complexity set of models you will get low training error, but you might cause the overfitting problem.

If you take a very simple set of models, you have low complexity, but the training error may increase.

5. Declaration

Most formulas and technological details in the report are quotations from these articles / papers listed in Section 7 References. It's inconvenient to mark every in-text direct or indirect quotation because there are too many quotations. Therefore, in the report I didn't use in-text quotation markings. Please refer to this declaration and Section 7 References.

6. Experiment

6.1 Summary

In the matlab source code, I implemented soft margin binary class linear SVMs by solving primal and dual problems, and use these trained classifiers to predict the test dataset. I tried 3 different C parameters 0.1, 1.1 and 10.1. I used "a1a" dataset on libsvm.

For each C parameter, I trained 2 SVM classifiers. One is from primal form, and the other is from dual form. Then I use them to perform prediction on training dataset and test dataset, and record some statistics like error rate. Moreover, I also compare these classifiers versus one another, and versus libsvm classifiers.

"a1a" training set is 1605 records * 123 features, and the test set is 30956 records * 123 features.

6.2 Statistics of experiments

	C=0.1	C=1.1	C=10.1
sum(abs(**w**(primal)-w(dual)))	3.7413e-05	0.0033	4.9065e-04
sum(abs(**w**(primal)-w(libsvm)))	0.0367	0.0734	0.1344
sum(abs(**w**(libsvm)-w(dual)))	0.0367	0.0715	0.1344
*b*_libsvm	-1.0361	-1.6482	-1.8314
*b*_primal	-1.0355	-1.6457	-1.8361
*b*_dual	-1.0339	-1.6374	-1.8361
Optimal_value calculated by libsvm	58.740388	593.159348	5260.285095
Optimal_value calculated by solving the primal	58.7404	593.159	5260.29
Optimal_value calculated by solving the dual	58.7404	593.159	5260.29
Duality gap = primal optimal value – dual optimal value	4.07e-07	1.41e-05	9.33e-05
Error rate, libsvm model on train set	15.2%	13.77%	13.09%
Error rate, primal classifier on train set	15.33%	13.77%	13.08%
Error rate, dual classifier on train set	15.26%	13.64%	13.08%
Error rate, libsvm model on test set	15.69%	16.16%	16.252%
Error rate, primal classifier on test set	15.68%	16.16%	16.24%
Error rate, dual classifier on test set	15.68%	16.17%	16.24%

6.3 Compare w, b obtained by primal and dual problems

From summing up the absolute value of $\Delta\, \mathbf{w}_i$ between w_primal and w_dual, we see the difference is very small. This sum of absolute values of w_difference is 3.7413e-05, 0.0033, 4.9065e-04 under different C values, very close to 0.

Therefore, we could claim that the **w** from primal problem and dual problem are almost the same.

From the statistic table, the b_primal and b_dual under each C value are also almost the same, very slight difference.

6.4 Check duality gap

The duality gap under C = 0.1, 1.1, 10.1 are 4.07e-07, 1.41e-05 and 9.33e-05. Again, almost equal to 0. This result proves that the strong duality holds in SVMs.

6.5 Compare my **w**, b and α with those of libsvm.

The α vector of libsvm is stored in libsvm_model.sv_coef vector, and corresponding support vectors' indices are stored in libsvm_model.sv_indices vector.

My α vector is calculated and stored in a vector called alphaV. Since the size of α vector is big, more than 500, it's not feasible to compare the elements in α vector one by one. Instead, the **w** vector is calculated based on α vector, and the size of **w** is relatively small. Hence I will compare **w** instead of comparing α vector.

w is a 119*1 (or 123*1 for test set) vector, how to measure how same two **w** vectors are. I choose to sum up the absolute value of $\Delta\, \mathbf{w}_i$ between mine and those of libsvm. From the statistics table, we could see that, the sum of abs ($\Delta\, \mathbf{w}_i$) between mine and those of libsvm range from 0.0367 to 0.1344. Considering the size of **w** vector is more than 100, the average difference of each element in these **w** vectors is very small.

Now let's consider the b variable, when C takes 0.1, 1.1 and 10.1. They are very close to each other.

*b*_libsvm	-1.0361	-1.6482	-1.8314
*b*_primal	-1.0355	-1.6457	-1.8361
*b*_dual	-1.0339	-1.6374	-1.8361

Therefore, we could claim that the difference between my parametres and libsvm parametres are very small.

6.6 Compare training and testing errors of my SVMs and libsvm

From the statistics table, all the SVM classifiers (mine and libsvm) behave better on training set than on test set. Under the same data set and the same C value, my SVM classifiers and libsvm have very close error rate.

I also found some interesting information here, which is, the error rate on training set decreases when C value increases from 0.1 to 10.1, whilst the error rate on test set increases under same situation, C from 0.1 to 10.1. The reason is exactly explained in Section 4. C value relates to the level of complexity of the models. Big C value means taking a high complexity set of models, then you will get low training error, but you might cause the overfitting problem on test set.

Small C value means a relatively simple set of models, you decrease overfitting so the error rate on test set is low, but the training error may increase.

6.7 Other comparisons

In the code, I also compare how many data points are predicted differently by my primal and dual classifier.

# of different predicted data points by 2 classifiers	C=0.1	C=1.1	C=10.1
On Train Set (1605 points)	1	2	0
On Test Set (30956 points)	11	32	8

We could infer from these information that, when C=10.1, the 2 classifiers are most close to each other, their “disagree” is least.

7. References

[1] Wikipedia,. 'Support Vector Machine'. N.p., 2015. Web. 29 Aug. 2015.
[2] Cs.columbia.edu,. (2015). Retrieved 29 August 2015, from http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf
[3] Forums.cs.adelaide.edu.au,. (2015). The University of Adelaide - Login. Retrieved 29 August 2015, from https://forums.cs.adelaide.edu.au/pluginfile.php/46083/mod_resource/content/1/L2%20Supervised%20Learning.pdf
[4] Web.mit.edu,. (2015). Retrieved 29 August 2015, from http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf
[5] Christopher J.C. Burges. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2:121-167, 1998
[6] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines, 2001. Softwareavailable at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[7] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A Practical Guide to Support Vector Classification, 2010.