1 Kernelized Perceptron

Given a set of training samples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}\ \ , y\ \ )$ where $y \in \{-1, 1\}$, the Perceptron algorithm learns a weight vector $\boldsymbol{w}$ by iterating through all training samples. For each $\boldsymbol{x}$, if the prediction is incorrect, we update $\boldsymbol{w}$ by $\boldsymbol{w} \leftarrow \boldsymbol{w} + y_i\boldsymbol{x}$. Now we would like to *kernelize* the Perceptron algorithm. Assume we map $\boldsymbol{x}$ to $\varphi(\boldsymbol{x})$ through a nonlinear feature mapping φ, and we want to learn a new weight vector $\boldsymbol{w}$ that makes prediction by $y = \text{sign}(\boldsymbol{w}^\top \varphi(\boldsymbol{x}))$.

(a) Show that $\boldsymbol{w}$ is always a linear combination of feature vectors, i.e. $\boldsymbol{w} = \sum_{=1} \alpha_i \varphi(\boldsymbol{x}\ )$.

(b) Show that while the update rule for $\boldsymbol{w}$ for a kernelized Perceptron does depend on the explicit feature mapping $\varphi(\boldsymbol{x})$, the prediction can be re-expressed and thus depends only on the inner products between nonlinear transformed features.

(c) Given the above, show that we do not need to explicitly store $\boldsymbol{w}$. Instead, we can implicitly use it by maintaining all the α_i. Please give the outline of the algorithm to achieve so. You should indicate how α_i is to be initialized, when to update α_i and which α_i is to be updated and how it is updated.

2 Support Vector Machine without Bias Term

Support vector machines (SVMs) learn the function $f(\boldsymbol{x}) = \boldsymbol{w}^\top \varphi(\boldsymbol{x}) + b$ where φ is a feature mapping, and predict the label of $\boldsymbol{x}$ using $y = \text{sign}(\boldsymbol{w}^\top \varphi(\boldsymbol{x}) + b)$. Now we consider a new model of SVM which does not use the bias term b, i.e. $f(\boldsymbol{x}) = \boldsymbol{w}^\top \varphi(\boldsymbol{x})$ and $y = \text{sign}(\boldsymbol{w}^\top \varphi(\boldsymbol{x}))$.

(a) Introduce slack variables $\{\xi_i\}$, and write down the primal form of the model for a training data set $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}\ \ , y\ \ )$

(b) Write down the Lagrangian of the primal form.

(c) Take derivatives with respect to the primal variables $\boldsymbol{w}$ and ξ_i and set them to zero. This gives rise to equations linking the primal variables and the dual variables. Write down these equations. Make sure you can express the primary variables in terms of the dual variables.

(d) Using the derived equations, substitute the primal variables into the Lagrangian. You then re-organize the Lagrangian and make sure the final term depends on the dual variables only (you might need a few equations in the last step to eliminate a few terms)

(e) Write down the dual form – it should be an optimization problem with proper constraints on the dual variables.

(f) Point out the main difference between the dual form in (e) and the dual form of original SVM (with bias term), described in the lecture.

(g) Show that the dual form in (e) is a convex optimization problem (in this particular case, you need to show the dual form is either maximizing a concave function or minimizing a convex function).

3 Sample questions for Quiz #1

3.1 Regularization

Consider a training data D used to adjust parameters θ of a model $L(D) = p(D;\theta)$, where $L(D)$ is the likelihood of the data. Normally, I will find the optimal θ by optimizing the likelihood. Now, I would like to use a L_2 norm based regularization to prevent overfitting. What does the objective function look like then? And should I minimize or maximize this objective function?

3.2 Logistic Regression

Consider a binary classification problem using multinomial logistic regression. Namely, each class of $\{0, 1\}$ has a weight vector $\boldsymbol{w}_0$ and $\boldsymbol{w}_1$. Let the input feature be $\boldsymbol{x}$ and the output of the classifier be y.

(a) Write down the posterior probability $p(y = 0|\boldsymbol{x})$ and $p(y = 1|\boldsymbol{x})$ in terms of these weights.

(b) Show that the posterior probability is unchanged if a constant vector $\boldsymbol{b}$ is added to both weight vectors.

(c) Observe that the log-likelihood is unchanged if a constant vector is added to both weight vectors. Therefore, the maximum conditional likelihood estimates of the parameters are not unique. A common technique to arrive at a unique solution is to use the regularized multinomial logistic regression. Assume you use the square of the weight vectors's L_2-norm (i.e. $\|\boldsymbol{w}\|_2^2$) as a form of regularization. Show that the regularized multinomial logistic regression has a unique solution, therefore, eliminating the invariance.

4 Programming

In this part, you will experiment with SVMs on a real-world dataset. You will implement linear SVM (i.e., SVM using the original features, namely the nonlinear mapping is the identity mapping) using Matlab. Also, you will use a widely used SVM toolbox called LIBSVM to experiment with kernel SVMs.

Dataset: We have provided the *Splice Dataset* from UCI's machine learning data repository (https://archive.ics.uci.edu/ml/datasets/Molecular+Biology+(Splice-junction+Gene+Sequences). The dataset is for a binary classification problem in biological field. The dimensionality of the input feature is 60, and the training and test sets contain 1,000 and 2,175 samples, respectively (they are provided in Blackboard as `splice_train.mat` and `splice_test.mat` which can be directly loaded into Matlab).

Please follow the steps below:

4.1 Data preprocessing

Preprocess the training and test data by

(a) computing the mean of each dimension and subtracting it from each dimension

(b) dividing each dimension by its standard deviation

Notice that the mean and standard deviation should be estimated from the training data and then applied to both datasets.

4.2 Implement linear SVM

Please fill in the Matlab function `trainsvm` in `trainsvm.m` and `testsvm.m` in `testsvm.m` (in Blackboard). The input of `trainsvm` contain training feature vectors and labels, as well as the tradeoff parameter C. The output of `trainsvm` contain the SVM parameters (weight vector and bias). In your implementation, you need to solve SVM in its primal form

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}} \ & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_i \xi_i \\ \text{s.t.} \ & y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i, \forall i \\ & \xi_i \geq 0, \forall i \end{aligned}$$

Please use `quadprog` function in Matlab to solve the above quadratic problem. For `testsvm`, the input contain testing feature vectors and labels, as well as SVM parameters; the output contains the test accuracy.

4.3 Cross validation for linear SVM

Use 5-fold cross validation to select the optimal C for your implementation of linear SVM.

(a) Report the cross-valiation accuracy (averaged accuracy over each validation set) and average training time (averaged over each training subset) on different C taken from $\{4^{-6}, 4^{-5}, \cdots, 4, 4^2\}$. How does the value of C affect the cross validation accuracy and average training time? Explain your observation.

(b) Which C do you choose based on the cross validation results?

(c) For the selected C, report the test accuracy.

4.4 Use linear SVM in LIBSVM

LIBSVM is widely used toolbox for SVM and has Matlab interface. Download LIBSVM from http://www.csie.ntu.edu.tw/~cjlin/libsvm/ and install it according to the README file (make sure to use Matlab interface provided in LIBSVM toolbox). For each C from $\{4^{-6}, 4^{-5}, \cdots, 4, 4^2\}$, apply 5-fold cross validation (use `-v` option in LIBSVM) and report the cross validation accuracy and average training time.

(a) Is the cross validation accuracy the same as that in 4.3? Note that LIBSVM solves linear SVM in dual form while your implementation does it in primal form.

(b) How faster is LIBSVM compared to your implementation, in terms of training?

4.5 Use kernel SVM in LIBSVM

LIBSVM supports a number of kernel types. Here you need to experiment with the polynomial kernel and RBF (Radial Basis Function) kernel.

(a) **Polynomial kernel**. Please tune C and `degree` in the kernel. For each combination of (C, `degree`), where $C \in \{4^{-3}, 4^{-4}, \cdots, 4^6, 4^7\}$ and `degree` $\in \{1, 2, 3\}$, report the 5-fold cross validation accuracy and average training time.

(b) **RBF kernel**. Please tune C and `gamma` in the kernel. For each combination of (C, `gamma`), where $C \in \{4^{-3}, 4^{-4}, \cdots, 4^6, 4^7\}$ and `gamma` $\in \{4^{-7}, 4^{-6}, \cdots, 4^{-1}, 4^{-2}\}$, report the 5-fold cross validation accuracy and average training time.

Based on the cross validation results of Polynomial and RBF kernel, which kernel type and kernel parameters will you choose? Report the corresponding test accuracy.

Submission Instruction: You need to provide the followings:

- Provide your answers to problems 1-3, 4.3, 4.4 and 4.5 in hardcopy. The papers need to be stapled and submitted to CS front desk.
- Submit BOTH the code and report via Blackboard. The only acceptable language is MATLAB.
- For your program, you must include the main function called `CSCI567_hw3.m` in the root of your folder. After running this main file, your program should be able to generate all of the results needed for this programming assignment (you can assume that LIBSVM will be added into the directory of your code, thus LIBSVM functions can be called directly). You can have multiple files (i.e. your sub-functions), however, the only requirement is that once we unzip your folder and execute your main file, your program should execute correctly. Please double-check your program before submitting. You should only submit one `.zip` file. Other file format is NOT allowed. Also, please name it as `[lastname]_[firstname]_hw3.zip`.

Collaboration You may collaborate. However, you need to write your own solution and submit separately. You also need to list with whom you have discussed.