1 Kernel Methods (15 points)

Answer the following questions regarding Kernel functions:

(a) Suppose you have a constant Kernel function, $k(\mathbf{x}_i, \mathbf{x}_j) = c$ for all $\mathbf{x}_i, \mathbf{x}_j$. Is k a symmetric positive semidefinite kernel?

(b) Assume that k_1 and k_2 are two symmetric, positive semidefinite kernels defined on the same space, let k be their minimum, $k(u, v) = min k_1(u, v), k_2(u, v)$. Is k also a symmetric positive semidefinite kernel?

(c) A symmetric, positive semidefinite kernel, k is defined on χ. Let K denote the kernel matrix corresponding to the set $x_1, ..., x_n \subset \chi$ (that is, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$).For $u \in \chi$, define $k_x(u) = (k(u, x_1), ..., k(u, x_n))$, and for $u, v \in X$, define $\tilde{k}(u, v) = k(u, v) - k_x(u)' K^{-1} k_x(v)$ assuming K has full rank. Is $\tilde{k}$ also a symmetric, positive semidefinite kernel?

2 Support Vector Machines (20 Points)

Let's formulate a support vector description of cluster and outliers in a dataset. Assume we have a dataset of $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_N\}$, each sample being a d dimensional raw feature vector. In this dataset, there is no label associated with the data. Nonetheless, some of the data might be *normal* (i.e. inside a cluster) and some of the data might be *abnormal* (i.e. outside of a cluster). Our approach builds on this intuition: normal data should "look like" each other and abnormal data should "look different" from normal data. Thus, to model "look like", we would like to say the transformation of a normal data point $\phi(\boldsymbol{x})$ should be close to a center.

This is described as a constraint:

$$||\phi(\boldsymbol{x}_j) - \boldsymbol{c}||^2 \leq R^2 \quad \forall j, \tag{1}$$

where $|| \cdot ||$ is the Euclidean norm, $\boldsymbol{c}$ is a center of the normal samples, and distance R represent the radius of cluster boundary from the center $\boldsymbol{c}$ – we do not know their value yet. To model "look different", we would like to say an abnormal data point $\phi(\boldsymbol{x})$ should be far to the center of all data points.

$$||\phi(\boldsymbol{x}_j) - \boldsymbol{c}||^2 \geq R^2 \quad \forall j. \tag{2}$$

We tend to believe all normal data points should be able to form a small circle. Thus we are interested in finding a small r as much as possible. However, a small circle will make many data points becoming *abnormal*. To balance these two conflicting forces, we minimize the following regularized objective function

$$\min_{R, \boldsymbol{c}, \boldsymbol{\xi}} \frac{1}{2} R^2 + c \sum_n \xi_n \tag{3}$$

$$\text{s.t.} ||\phi(\boldsymbol{x}_n) - \boldsymbol{c}||_2^2 \leq R^2 + \xi_n, \quad \forall n \tag{4}$$

$$\xi_n \geq 0 \tag{5}$$

where ξ_n is the slack variable for the n-th sample. A larger ξ_n implies that $\boldsymbol{x}_n$ is very far away from the center. C is regularization parameter/coefficient. The larger the C is, the more unlikely we are to make data points as *abnormal*. In other words, C controls the ratio of how many data points will become regarded as *normal*.

The above formulation is in the primal form as we are using the transformed feature $\phi(\boldsymbol{x})$. Note that the formulation looks like a SVM. Please derive the corresponding dual formulation, just like we have done for SVM. You should:

- Give the dual optimization problem in Lagrange multipliers. Your dual formulation will depend on inner products between transformed features. Then, rewrite your dual formulation using a kernel function $K(\cdot,\cdot)$.
- You should also give the solution to the primal variables r and c in terms of these Lagrange multipliers.
- Also write down the decision rule on how to determine whether x is *normal* or *abnormal*.

Note that all these should be given in terms of the kernel function, instead of the transformed features.

3 Boosting (Total: 30 points, Bonus: 15 points)

(a) **(Exponential Loss and Logistic Loss: (15 points))** Consider the following binary classification problem with exponential loss (6) and log loss (7)

$$\mathcal{L}_e(f) := E_{x,y}[\exp(-yf(x))] = \frac{1}{N}\sum_{n=1}^{N}\exp(-y_i f(x_i)) \tag{6}$$

$$\mathcal{L}_\sigma(f) := E_{x,y}\log[1+exp(-2yf(x))] = \frac{1}{N}\sum_{n=1}^{N}\log[1+\exp(-2y_i f(x_i))] \tag{7}$$

where $y \in \{-1,+1\}$ is a binary class label.

- **(Similarity:)** Show that for both loss functions, the conditional distribution of y is

$$P(y|x) = \frac{\exp(f(x))}{\exp(f(x)) + \exp(-f(x))}$$

(**Hint**) In the case of exponential loss, you may use *variational minimization* of the loss function w.r.t. $f(x)$ to obtain the expression of optimal $f^*(x)$. You do not have to understand the *calculus of variations*, the mathematical foundation of *variational minimization*. Instead, you can simply consider function $f(x)$ as a parameter and tread it the same way to parameters in MLE, e.g. $\frac{\partial\mathcal{L}(\omega)}{\omega} = 0$.

Note: Although $P(y|x)$ as a function of $f(x)$ has the same form for both loss, the function $f(x)$ learned from different losses are different.

- **(Difference:)** Show that the log loss can be considered as negative log-likelihood, while the exponential loss cannot.

  (**Hint**:) Respectively, show that $\log(1 + \exp(-2yf))$ and $\exp(-yf)$ can and cannot be written as logarithm of probability functions for binary variable $y \in \{-1, +1\}$.

(b) **(Boosting using log loss:) (Bonus: 15 points)** In the class we have studied how to learn the boosting classifier $f(x) = \sum_l \alpha_l f_l(x)$ by optimizing the exponential loss. In this problem, let's study the learning of a boosting classifier $f(x) = \sum_l f_l(x)$ under log loss.

Before we optimize $f(x)$, let's recap the Newton's method in learning a logistic regression model $P(y|\boldsymbol{x}) = \frac{1}{1+\exp(-2(\theta\cdot\boldsymbol{x})y)}$. Consider making a second-order Taylor series approximation of $\mathcal{L}(\theta)$ around θ_k, the learned parameter at the end of k-th iteration,

$$\mathcal{L}_{quad}(\theta) = \mathcal{L}(\theta_k) + \boldsymbol{g}_k^T(\theta - \theta_k) + \frac{1}{2}(\theta - \theta_k)^T\boldsymbol{H}_k(\theta - \theta_k) \tag{8}$$

Let's rewrite it as

$$\mathcal{L}_{quad}(\theta) = \theta^T\boldsymbol{A}\theta + \boldsymbol{b}^T\theta + c, \text{ where} \tag{9}$$

$$\boldsymbol{A} = \frac{1}{2}\boldsymbol{H}_k \tag{10}$$

$$\boldsymbol{b} = \boldsymbol{g}_k - \boldsymbol{H}_k\theta_k \tag{11}$$

The minimum of $\mathcal{L}_{quad}$ is considered as the parameter learned at the end of $(k+1)$-th iteration:

$$\theta_{k+1} = -\frac{1}{2}\boldsymbol{A}^{-1}b = \theta_k - \boldsymbol{H}_k^{-1}\boldsymbol{g}_k \tag{12}$$

This is the parameter updating rule of Newton's method.

Now let's return to boosting function $f(x)$. Define $f_m(x) := \sum_{l=1}^m \phi_l(x)$, the summation of m base functions $\{\phi_1(x)\cdots\phi_m(x)\}$. Consider adding the $(m+1)$-th base function, $f_{m+1}(x) = f_m(x) + \phi_{m+1}(x)$:

- By making an analogy between f_m and θ_m, prove that given $f_m(x)$, the optimization problem w.r.t the $(m+1)$-th base function, $\arg\min_{\phi_{m+1}(x)} \mathcal{L}_\sigma(f_{m+1})$ can be approximated by a **weighted least square problem** $\arg\min_{\phi(x)} \sum_{n=1}^N \omega_n^{(m)}(z_n^{(m)} - \phi(x_n))^2$. In this proof, you need to figure out the expression for weight coefficient for n-th sample $\omega_n^{(m)}$, and working response variable $z_n^{(m)}$. Both of them are dependent on the outcome of $f_m(x)$ model.

Now you have derived the model of **LogitBoost**.

4 Bias-Variance Trade-off (Bonus: 15 points)

Consider a dataset with n data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$, drawn from the following linear model:

$$y = \mathbf{x}^\top\boldsymbol{\beta}^\star + \varepsilon,$$

where ε is a Gaussian noise and the star sign $(*)$ is used to differentiate the true parameter from the estimators that will be introduced later. Consider the L_2 regularized linear regression as follows:

$$\widehat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\},$$

where $\lambda \geq 0$ is the regularization parameter. Let $X \in \mathbb{R}^{n \times p}$ denote the matrix obtained by stacking $\mathbf{x}_i^\top$ in each row.

(a) Find the closed form solution for $\widehat{\boldsymbol{\beta}}_\lambda$ and its distribution.

(b) Calculate the bias term $\mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda] - \mathbf{x}^\top \boldsymbol{\beta}^\star$ as a function of λ and some feature vector $\mathbf{x}$.

(c) Calculate the variance term $\mathbb{E}\left[\left(\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda - \mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda]\right)^2\right]$ as a function of λ and some feature vector $\mathbf{x}$.

(d) Use the results from parts (b) and (c) and the bias–variance theorem to analyze the impact of λ in the squared error. (i.e. which term dominates when λ is small or large?)

Programming Questions

5 Support Vector Machines (50 points)

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 *Diabetic Retinopathy Debrecen Data Set* from UCI's machine learning data repository (http://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set. This dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not. The dimensionality of the input feature is 19, and the training and test sets contain 768 and 383 samples, respectively (they are provided in Blackboard as `diabetic-train.mat` and `diabetic-test.mat` which can be directly loaded into Matlab).

5.1 Data preprocessing

All the features in the datasets are numerical. Please scale every feature into range $[0, 1]$ independently. Remember that you need to use the same scaling parameters for training and testing data.

5.2 Implement linear SVM

Please write Matlab functions `trainsvm` as `trainsvm.m` and `testsvm` as `testsvm.m`. 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

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

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.

5.3 Cross validation for linear SVM

Use 3-fold cross validation to select the optimal C for your implementation of linear SVM.

(a) Report the 3-fold 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) You need to write a script `own_linear.m` using your `trainsvm`, `testsvm` and the optimal C selected from cross validation. Your script should train a linear SVM with selected C and output the accuracy on the test set.

5.4 Use linear SVM in LIBSVM

LIBSVM is widely used toolbox for SVM and has Matlab interface. Download LIBSVM from `https://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 3-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 5.3? 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?

5.5 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 $C \in \{4^{-3}, 4^{-2}, \cdots, 4^6, 4^7\}$ and `degree` $\in \{1, 2, 3\}$, report the 3-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 $C \in \{4^{-3}, 4^{-2}, \cdots, 4^6, 4^7\}$ and `gamma` $\in \{4^{-7}, 4^{-6}, \cdots, 4^{+1}, 4^{+2}\}/\bar{d}$, **report** the 3-fold cross validation accuracy and average training time.

The variable $\bar{d}$ in `gamma` hyperparameters is the **median** of pairwise square distances of training samples (e.g. suppose we have 10 training samples, there are 90 pairwise square distances, the median of these 90 distances is $\bar{d}$). We provided a function `distMetric.m` to calculate the median distance. You need to figure out how to call this function.

Based on the cross validation results of Polynomial and RBF kernel, which kernel type and kernel parameters will you choose? You need to write a script `libsvm.m` using the best kernel and the parameters selected from cross-validation. Your script should train a SVM using libsvm with selected kernel and parameters and output the accuracy on the test set.

6 Bias/Variance Trade-off (Bonus: 20 points)

Let the function $f(x) = 2x^2 + \epsilon$, where ϵ is Gaussian noise drawn from $\mathcal{N}(0, 0.1)$. We are also given the following functions:

- $g_1(x) = 1$
- $g_2(x) = w_0$
- $g_3(x) = w_0 + w_1 x$
- $g_4(x) = w_0 + w_1 x + w_2 x^2$
- $g_5(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3$
- $g_6(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4$

(a) Randomly generate 100 datasets. Each dataset contains 10 samples (x_i, y_i), where x_i is uniformly sampled from $[-1, 1]$ and $y_i = f(x_i)$. For a given g function, estimate its parameters using linear regression and compute the sum-square-error on every dataset. Then plot the histogram of the mean-squared-error. Repeat this for each g function. Please provide the 6 histogram plots in the report. For each g function, estimate its bias^2 and variance, and report the results.

(b) In (a), change the sample size in each dataset to 100, and repeat the procedures. Plot the resulting histograms in the report. For each g function, estimate its bias^2 and variance, and report the results.

(c) Given your results in (a) and (b), discuss how the model complexity and sample size affect the bias^2 and variance.

(d) Consider function $h(x) = w_0 + w_1 x + w_2 x^2 + \lambda(w_0^2 + w_1^2 + w_2^2)$ where $\lambda \geq 0$ is a regularization parameter. Following (b) (i.e. 100 samples per dataset), estimate the bias^2 and variance of $h(x)$ when λ set to $0.01, 0.1, 1, 10$ respectively. Discuss how λ affect the bias^2 and variance.

Submission Instruction: You need to provide the followings:

- Provide your answers in PDF file, named as `CSCI567_hw3_spring16_yourUSCID.pdf`. You need to submit the homework in both hard copy (at CS567 Homework lockers by 4 pm of the deadline date) and electronic version as pdf file on Blackboard. If you choose handwriting instead of typing all the answers, 40% points will be deducted.
- Submit ALL the code and report via Blackboard. The only acceptable language is MATLAB. For your program, you MUST include the main function called `CSCI567_hw3_spring16.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 the programming assignment, either as plots or console outputs. 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. No other formats are allowed except `.zip` file. Also, please name it as `[lastname]_[firstname]_hw3_spring16.zip`.

Collaboration: You may collaborate. However, collaboration has to be limited to discussion only and you need to write your own solution and submit separately. You also need to list with whom you have discussed.