Introduction to Machine Learning **Fall Semester**

Homework 4: December 2, 2020

Due: December 16, 2020

Theory Questions

1. **(15 points) SGD with projection.** In the context of convex optimization, sometimes we would like to limit our solution to a convex set $\mathcal{K} \subseteq \mathbb{R}^d$; that is,

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{x} \in \mathcal{K} \end{aligned}$$

for a convex function f and a convex set $\mathcal{K}$. In this scenario, each step in the gradient descent algorithm might result in a point outside $\mathcal{K}$. Therefore, we add an additional projection step. The projection operator finds the closest point in the set, i.e.:

$$\Pi_{\mathcal{K}}(\mathbf{y}) := \arg\min_{\mathbf{x}\in\mathcal{K}} \|\mathbf{x} - \mathbf{y}\|$$

A modified iteration in the *gradient descent with projection* therefore consists of:

$$\begin{aligned} \mathbf{y}_{t+1} &= \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t) \\ \mathbf{x}_{t+1} &= \Pi_{\mathcal{K}}(\mathbf{y}_{t+1}) \end{aligned}$$

(a) Suppose we want to solve the SVM primal problem, as formulated in the context of SGD:

$$\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m}\sum_{i=1}^{m} \max(0, 1 - y_i \mathbf{w}\cdot\mathbf{x}_i)$$

However, we want to find a solution with a bounded norm; i.e. $\mathbf{w} \in \mathcal{K}$, where $\mathcal{K} = \{\mathbf{x} | \|\mathbf{x}\| \leq R\}$. Modify the SGD algorithm to include a projection step. How do you calculate the projection?

(b) Let $\mathcal{K}$ be a convex set, $\mathbf{y} \in \mathbb{R}^d$ and $\mathbf{x} = \Pi_{\mathcal{K}}(\mathbf{y})$. Prove that for any $\mathbf{z} \in \mathcal{K}$, we have $\|\mathbf{y} - \mathbf{z}\| \geq \|\mathbf{x} - \mathbf{z}\|$.

(c) Prove that the convergence theorem for GD still holds.

2. **(15 points) SVM with multiple classes.** One limitation of the standard SVM is that it can only handle binary classification. Here is one extension to handle multiple classes. Let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and now let $y_1, \ldots, y_n \in [K]$, where $[K] = \{1, 2, \ldots, K\}$. We will find a separate classifier $\mathbf{w}_j$ for each one of the classes $j \in [K]$, and we will focus on the case of no bias ($b = 0$). Define the following loss function (known as the multiclass hinge-loss):

$$\ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i) = \max_{j\in[K]} \left(\mathbf{w}_j \cdot \mathbf{x}_i - \mathbf{w}_{y_i} \cdot \mathbf{x}_i + \mathbb{1}(j \neq y_i)\right)$$

Define the following multiclass SVM problem:

$$f(\mathbf{w}_1, \ldots, \mathbf{w}_K) = \frac{\beta}{2}\sum_{j\in[K]} \|\mathbf{w}_j\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{w}_1, \ldots, \mathbf{w}_K, \mathbf{x}_i, y_i)$$

After learning all the $\mathbf{w}_j$, $j \in [K]$, classification of a new point $\mathbf{x}$ is done by $\arg\max_{j\in[K]} \mathbf{w}_j \cdot \mathbf{x}$. The rationale of the loss function is that we want the "score" of the true label, $\mathbf{w}_{y_i} \cdot \mathbf{x}_i$, to be larger by at least 1 than the "score" of each other label, $\mathbf{w}_j \cdot \mathbf{x}_i$. Therefore, we pay a loss if $\mathbf{w}_{y_i} \cdot \mathbf{x}_i - \mathbf{w}_j \cdot \mathbf{x}_i \leq 1$, for $j \neq y_i$.

(a) **(7 points)** Show that when $K = 2$, f reduces to the standard SVM with binary classification. That is, denote by $\mathbf{w}_1^*(\beta), \mathbf{w}_2^*(\beta)$ the solution of the multiclass problem with $K = 2$, with a penalty β. Denote by $\mathbf{w}^*(\beta')$ best solution of the standard SVM, with a penality β':

$$\frac{\beta'}{2}\|\mathbf{w}\|^2 + \frac{1}{n}\sum_{i=1}^{n}\max(0, 1 - y_i\mathbf{w}\cdot\mathbf{x}_i)$$

Show that for every $\beta > 0$, there is a $\beta' > 0$, so that the multiclass classifer defined by $\mathbf{w}_1^*(\beta), \mathbf{w}_2^*(\beta)$ and the standard SVM classifier defined by $\mathbf{w}^*(\beta')$ are the same. Show the correspondence β and β', and between $\mathbf{w}_1^*(\beta), \mathbf{w}_2^*(\beta)$ and $\mathbf{w}^*(\beta')$ (i.e., an equation showing the relationship between them).

(Hint: First show that the solution to the multiclass SVM satisfies $\mathbf{w}_1^*(\beta) = -\mathbf{w}_2^*(\beta)$. Use the fact that if $\mathbf{u}_1 - \mathbf{u}_2 = \mathbf{v}_1 - \mathbf{v}_2$ then $\ell(\mathbf{u}_1, \mathbf{u}_2, \mathbf{x}_i, y_i) = \ell(\mathbf{v}_1, \mathbf{v}_2, \mathbf{x}_i, y_i)$.)

(b) **(4 points)** Like the hinge loss, the multi-class hinge-loss is not differentiable everywhere. Identify the non-differentiable points, and calculate the gradient with respect $\mathbf{w}_j$, $j \in [K]$, for the differentiable points. Use it to derive a Stochastic (Sub)gradient Descent algorithm (it is a sub-gradient algorithm because of the non-differentiability at certain points) to find $\mathbf{w}_j, j \in [K]$ that minimize f (use a constant step size, η). Write a pseudo-code for the algorithm.

(c) **(4 points)** Assume $\beta = 0$. Consider the case where the data is linearly separable. Namely, there exists $\mathbf{w}_1^*, ..., \mathbf{w}_K^*$ such that $y_i = argmax_y \mathbf{w}_y^* \cdot \mathbf{x}_i$. Show that any minimizer of $f(\mathbf{w}_1, ..., \mathbf{w}_K)$ will have zero classification error.

3. **(15 points) Kernel Implementation of Primal SGD.** In Programming Assignment 2 in Exercise 3 you implemented the SGD update for regularized hinge loss minimization (binary classification):

$$w_{t+1} = (1-\eta)w_t + b_t\eta C y_i x_i,$$

where $b_t = 1$ if $y_i w_t \cdot x_i > 1$ and 0 else. For the rest of this exercise you can assume that $b_t = 1$ (although of course that is not generally true). Now assume that instead of x_i you want to implement this algorithm for a feature $\phi(x)$, where ϕ may be infinite dimensional. Also, assume you have a kernel function $K(x, x') = \phi(x) \cdot \phi(x')$ that can evaluated efficiently. The SGD update you want to perform is (assume fixed step size η):

$$w_{t+1} = (1-\eta)w_t + b_t\eta C y_i \phi(x_i).$$

This seems impossible to implement because ϕ is infinite dimensional. But, show that you can use the kernel trick to implement each update in $O(n)$ time and $O(n)$ storage, where n is the number of data points (ignoring the cost of the call to the kernel function, and the cost of storing the input data points). Explain clearly how you are representing w using α coefficients and how these are updated.

4. **(10 points) Weighted SVM.** The soft margin SVM algorithm seen in class assumes that all data points are equally important. In this question we assume that every data point x_i

has a weight $0 \leq v_i \leq 1$ defining its importance. Thus, the corresponding SVM optimization problem takes the following form:

$$\begin{aligned}
\min_{w,b,\boldsymbol{\xi}} \quad & 0.5\|w\|_2^2 + C\sum_{i=1}^{n} v_i\xi_i \\
\text{s.t.} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i \qquad \forall i = 1, \ldots, n \\
& \xi_i \geq 0
\end{aligned}$$

(a) What is the Lagrangian?

(b) What is the dual problem?

(c) Assuming we solved the dual problem, how can we compute w, b, ξ?

5. **(10 points) All points are support vectors.** Let $S = \{x_i, y_i\}_{i=1}^{2n}$ be a training set such that $x_i \in \mathbb{R}^2$ and $y_i \in \{-1, 1\}$. Assume that the training points have the following form:

$$x_i = \begin{cases} (1, i) & i \leq n \\ (-1, i-n) & i > n \end{cases} \qquad y_i = \begin{cases} 1 & i \leq n \\ -1 & i > n \end{cases}$$

Solve the SVM optimization problem directly for this case, and show that every data point is a support vector.

6. **(15 points) Separability Using RBF Kernels.** Consider the RBF kernel with $\sigma = 1$, and scalar inputs $x \in \mathbb{R}$. Assume you are given a training set $\{(x_i, y_i)\}_{i=1}^{n}$ where all x_i are different.

(a) Show that hard SVM with the RBF kernel will always achieve zero error. You can use the following facts:

i. For any $\alpha_1 \neq \alpha_2 \ldots \neq \alpha_n$ and $\beta_1 \neq \beta_2 \ldots \neq \beta_n$ define the matrix $A_{ij} = e^{\alpha_i \beta_j}$ (this is called a generalized Vandermonde matrix). Then A is full rank.

ii. If the kernel matrix for the training data is full rank then hard SVM will achieve zero training error (see lecture slides).

(b) Show that this result does not hold if there is $x_1 = x_2$.

Programming Assignment

Submission guidelines:

- Download the supplied files from Moodle. Details on every file will be given in the exercises. You need to update the code only in the skeleton files, i.e., the files that have a prefix "skeleton". Written solutions, plots and any other non-code parts should be included in the written solution submission.
- Your code should be written in Python 3.
- Make sure to comment out or remove any code which halts code execution, such as matplotlib popup windows.
- Your code submission should include the file `svm.py`.

1. **(25 points) SVM.** In this exercise, we will explore different kernels for SVM and the relation between the parameters of the optimization problem. We will use an existing implementation of SVM: the SVC class from sklearn.svm. This class solves the soft-margin SVM problem. You can assume that the data we will use is separable by a linear separator (i.e. that we could formalize this problem as a hard-margin SVM). In the file skeleton `svm.py` you will find two implemented methods:

   - `get_points` - returns training and validation sets of points in 2D, and their labels.
   - `create_plot` - receives a set of points, their labels and a trained SVM model, and creates a plot of the points and the separating line of the model. The plot is created in the background using matplotlib. To show it simply run `plt.show()` after calling this method.

   In the following questions, you will be asked to implement the other methods in this file, while using `get_points` and `create_plot` that were implemented for you.

   (a) **(5 points)** Implement the method `train_three_kernels` that uses the training data to train 3 kernel SVM models - linear, quadratic and RBF. For all models, set the penalty constant C to 1000.
   - How are the 3 separating lines different from each other? You may support your answer with plots of the decision boundary.
   - How many support vectors are there in each case?

   (b) **(10 points)** Implement the method `linear_accuracy_per_C` that trains a linear SVM model on the training set, while using cross-validation on the validation set to select the best penalty constant from $C = 10^{-5}, 10^{-4}, \dots, 10^{5}$. Plot the accuracy of the resulting model on the training set and on the validation set, as a function of C.
   - What is the best C you found?
   - Explain the behavior of error as a function of C. Support your answer with plots of the decision boundary.

   (c) **(10 points)** Implement the method `rbf_accuracy_per_gamma` that trains an RBF SVM model on the training set, while using cross-validation on the validation set to select the best coefficient $\gamma = 1/\sigma^2$. Start your search on the log scale, e.g., perform a grid search $\gamma = 10^{-5}, 10^{-4}, \dots, 10^{5}$, and increase the resolution until you are satisfied. Use $C = 10$

as the penalty constant for this section. Plot the accuracy of the resulting seperating line on the training set and on the validation set, as a function of γ.

- What is the best γ you found?
- How does γ affect the decision rule? Support your answer with plots of the decision boundary.