# 10-701 Introduction to Machine Learning

## Homework 3, version 1.4

*Due Oct 30, 11:59 am*

## Rules:

1. Homework submission is done via CMU Autolab system. Please package your writeup and code into a zip or tar file, *e.g.*, let `submit.zip` contain `writeup.pdf` and your code. Submit the package to `https://autolab.cs.cmu.edu/courses/10701-f15`.
2. Like conference websites, repeated submission is allowed. Please feel free to refine your answers since we will only grade the latest version. Submitting incomplete solutions early will be helpful in preventing last minute panic as well.
3. Autolab may allow submission after the deadline, note however it is because of the late day policy. Please see course website for policy on late submission.
4. We recommend that you typeset your homework using appropriate software such as LaTeX. If you are writing please make sure your homework is cleanly written up and legible. The TAs will not invest undue effort to decrypt bad handwriting.
5. You are allowed to collaborate on the homework, but you should write up your own solution and code. Please indicate your collaborators in your submission.

## 1 Neural Networks (50 Points) (Zhiting)

### 1.1 Neural network for regression

Figure.1 shows a two-layer neural network which learns a function $f : X \to Y$ where $X = (X_1, X_2) \in \mathbb{R}^2$. The weights $\boldsymbol{w} = \{w_1, \ldots, w_6\}$ can be arbitrary. There are two possible choices for the function implemented by each unit in this network:

- S: signed sigmoid function $S(a) = sign[\sigma(a) - 0.5] = sign[\frac{1}{1+\exp\{-a\}} - 0.5]$
- L: linear function $L(a) = ca$

where in both cases $a = \sum_i w_i X_i$.

1. Assign proper activation functions (S or L) to each unit in Figure.1 so this neural network simulates a linear regression: $Y = \beta_1 X_1 + \beta_2 X_2$.

   Answer:

   L, L, L

2. Assign proper activation functions (S or L) for each unit in Figure.1 so this neural network simulates a binary logistic regression classifier: $Y = \arg\max_y P(Y = y|X)$, where $P(Y = 1|X) = \frac{\exp(\beta_1 X_1 + \beta_2 X_2)}{1+\exp(\beta_1 X_1 + \beta_2 X_2)}$, and $P(Y = -1|X) = \frac{1}{1+\exp(\beta_1 X_1 + \beta_2 X_2)}$. Derive $\beta_1$ and $\beta_2$ in terms of $w_1, \ldots, w_6$.

   Answer:

   L, L, S

   $\beta_1 = c(w_1 w_5 + w_2 w_6)$

   $\beta_2 = c(w_3 w_5 + w_4 w_6)$

3. Assign proper activation functions (S or L) to each unit in Figure.1 so this neural network simulates a boosting classifier which combines two logistic regression classifiers, $f_1 : X \to Y_1$ and $f_2 : X \to Y_2$, to produce its final prediction: $Y = sign[\alpha_1 Y_1 + \alpha_2 Y_2]$. Use the same distribution in problem 1.1.2 for $f_1$ and $f_2$. Derive $\alpha_1$ and $\alpha_2$ in terms of $w_1, \ldots, w_6$.

   Answer:

   S, S, S

   $\alpha_1 = w_5$

   $\alpha_2 = w_6$

### 1.2 Convolutional neural networks

1. Count the total number of parameters in LeNet (pp.46, slides of Lecture.8). How many parameters in all of the convolutional layers? How many parameters in all of the fully-connected layers?

   Note:

   (a) The filter size of each convolutional and pooling(subsampling) layer:
   C1: $5 \times 5$ (i.e., each unit of C1 has a $5 \times 5$ receptive field in its preceding layer);
   S2: $2 \times 2$;
   C3: $5 \times 5$;
   S4: $2 \times 2$;

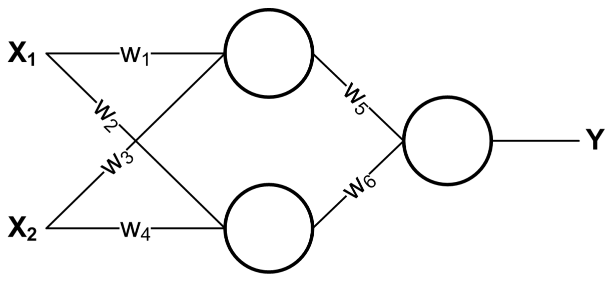

Figure 1: A two-layer neural network.

(b) Fully-connected layers in LeNet include C5, F6, and OUTPUT

Answer:

ConvLayers

C1: (5 * 5 + 1) * 6 = 156

S2: 0 or 2 * 6 = 12

C3: (5 * 5 * 3 + 1) * 6 + (5 * 5 * 4 + 1) * 9 + (5 * 5 * 6 + 1) = 1516
(Or: (16 * (6 * 5 * 5 + 1)) = 2416)

S2: 0 or 2 * 16 = 32

FCLayers

C5: (5 * 5 * 16 + 1) * 120 = 48120

F6: (120 + 1) * 84 = 10164

OUTPUT: (84 + 1) * 10 = 850

2. In a convolutional layer the units are organized into planes, each of which is called a feature map. The units within a feature map (indexed $q$) have different inputs, but all share a common weight vector, $\boldsymbol{w}^{(q)}$. A convolutional network is usually trained through backprorogation. Let $J^{(q)}$ be the number of units in the $q$th feature map, $z_j^{(q)}$ the activation of the $j$th unit, $x_{ji}^{(q)}$ the $i$th input for the $j$th unit, $w_i^{(q)}$ the $i$th element of $\boldsymbol{w}^{(q)}$, $L$ the training loss. Derive the gradient of $w_i^{(q)}$.

Answer:

$$\frac{\partial L}{\partial w_i^{(m)}} = \sum_j \frac{\partial L}{\partial z_j^{(m)}} \frac{\partial z_j^{(m)}}{\partial w_i^{(m)}} = \sum_j \delta_j^{(m)} x_{ji}^{(m)}$$

## 1.3 Gradient vanishing/explosion

In this problem we will study the difficulty of back-propagation in training deep neural networks. For simplicity, we consider the simplest deep neural network: one with just a single neuron in each layer, where the output of the neuron in the $j$th layer is $z_j = \sigma(a_j) = \sigma(w_j z_{j-1} + b_j)$. Here $\sigma$ is some activation function whose derivative on $x$ is $\sigma'(x)$. Let $m$ be the number of layers in the neural network, $L$ the training loss.

1. Derive the derivative of $L$ w.r.t. $b_1$ (the bias of the neuron in the first layer).

   Answer:
   $\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_m}\sigma'(a_1)\prod_{k=2}^{m}\sigma'(a_k)w_k$

2. Assume the activation function is the usual sigmoid function $\sigma(x) = 1/(1+\exp\{-x\})$. The weights $\boldsymbol{w}$ are initialized to be $|w_j| < 1$ $(j = 1, \ldots, m)$.

   (a) Explain why the above gradient $(\partial L/\partial b_1)$ tends to vanish $(\rightarrow 0)$ when $m$ is large.

   Answer:
   The derivative of sigmoid function reaches a maximum at $\sigma'(0) = \frac{1}{4}$. Since $|w_j| < 1$, we have $|w_j\sigma'(a_j)| < \frac{1}{4} < 1$

   (b) Even if $|w|$ is large, the above gradient would also tend to vanish, rather than explode $(\rightarrow \infty)$. Explain why. (A rigorous proof is not required.)

   Answer:
   To avoid the vanishing gradient problem we need $|w\sigma'(a)| \geq 1$. But, the $\sigma'(a)$ term also depends on $w : \sigma'(a) = \sigma(wz + b)$. If we make $w$ large we tend to make $wz + b$ very large, and $\sigma'(a)$ very small.

3. One of the approaches to (partially) address the gradient vanishing/explosion problem is to use the rectified linear (ReL) activation function instead of the sigmoid. The ReL activation function is $\sigma(x) = \max\{0, x\}$. Explain why ReL can alleviate the gradient vanishing problem as faced by sigmoid.

   Answer:
   As long as $a > 0, \sigma'(a) = 1$. So we don't have the issue as in 2(b).

4. A second approach to (partially) address the gradient vanishing/explosion problem is layer-wise pre-training. Restricted Boltzemann machine (RBM) is one of the widely-used models for layer-wise pre-training. Figure 2 shows an example of RBM which includes $K$ hidden units $\boldsymbol{h}$, and $J$ input units $\boldsymbol{v}$. Let us define the joint distribution as the following general form:

$$P(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z}\exp\left(\sum_i \theta_i\phi_i(\boldsymbol{v}, \boldsymbol{h})\right), \tag{1}$$

   where $Z = \sum_{\boldsymbol{v},\boldsymbol{h}} \exp\left(\sum_i \theta_i\phi_i(\boldsymbol{v}, \boldsymbol{h})\right)$ is the normalization term; $\phi_i(\boldsymbol{v}, \boldsymbol{h})$ are some features; $\theta_i$ are the parameters corresponding to the weights in the RBM. Consider the simplest learning algorithm, gradient descent. Show that

$$\frac{\partial \log P(\boldsymbol{v})}{\partial \theta_i} = \sum_{\boldsymbol{h}} \phi_i(\boldsymbol{v}, \boldsymbol{h})P(\boldsymbol{h}|\boldsymbol{v}) - \sum_{\boldsymbol{v},\boldsymbol{h}} \phi_i(\boldsymbol{v}, \boldsymbol{h})P(\boldsymbol{v}, \boldsymbol{h}). \tag{2}$$

   Answer:

We have that:

$$
\begin{aligned}
P(\boldsymbol{x},\boldsymbol{y}) &= \frac{1}{\sum_{\boldsymbol{x},\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right)} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right) \Rightarrow \\
P(\boldsymbol{x}) &= \frac{1}{\sum_{\boldsymbol{x},\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right)} \sum_{\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right) \Rightarrow \\
\log P(\boldsymbol{x}) &= \log \sum_{\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right) - \log \sum_{\boldsymbol{x},\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right) \Rightarrow \\
\frac{\partial \log P((x))}{\partial \theta_l} &= \sum_{\boldsymbol{y}} \frac{\exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right)}{\sum_{\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right)} \phi_l(\boldsymbol{x},\boldsymbol{y}) - \sum_{\boldsymbol{x},\boldsymbol{y}} \frac{\exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right)}{\sum_{\boldsymbol{x},\boldsymbol{y}} \exp\left(\sum_k \theta_k \phi_k(\boldsymbol{x},\boldsymbol{y})\right)} \phi_l(\boldsymbol{x},\boldsymbol{y}), \\
&= \sum_{\boldsymbol{y}} \frac{P(\boldsymbol{x},\boldsymbol{y})}{P(\boldsymbol{x})} \phi_l(\boldsymbol{x},\boldsymbol{y}) - \sum_{\boldsymbol{x},\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{y}) \phi_l(\boldsymbol{x},\boldsymbol{y}), \\
&= \sum_{\boldsymbol{y}} P(\boldsymbol{y} \mid \boldsymbol{x}) \phi_l(\boldsymbol{x},\boldsymbol{y}) - \sum_{\boldsymbol{x},\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{y}) \phi_l(\boldsymbol{x},\boldsymbol{y}).
\end{aligned}
$$

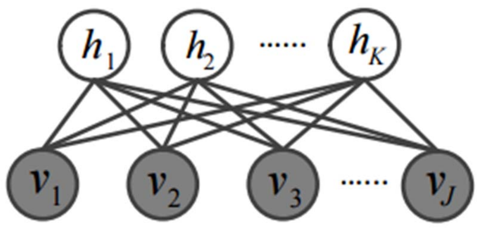

Figure 2: A restricted Boltzmann machine.

## 2 Support Vector Machines (50 Points) (Yuntian)

### 2.1 Support Vector Regression (25 Points)

We now extend support vector machines (SVM) to regression problems. Recall that in regression problems, we have $n$ data points $(x_i, y_i)_{i=1}^n$ where $x_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$. Given a function class $\mathcal{F}$ (*e.g.* linear or quadratic functions), we want to fit a function $f \in \mathcal{F}$ on the training set:

$$f^\star = \operatorname*{argmin}_{f \in \mathcal{F}} C \sum_{i=1}^n l(f(x_i), y_i) + R(f) \tag{3}$$

where $l(\cdot,\cdot)$ is the loss function, $R(f)$ is the regularization term, $C$ controls the regularization strength. The first part tries to fit data, and the second part penalizes complex $f$ to avoid over-fitting.

In the support vector regression (SVR) framework, we consider linear function class $\mathcal{F} = \{x \to w^T x\}$ (we do not consider interception term for simplicity). We use $\ell_2$-regularizer $R(f) = \frac{1}{2}\|w\|_2^2$ for $f(x) = w^T x$. For the loss function $l$, similar to the hinge-loss function in SVM classification, we employ an $\epsilon$-insensitive error function

$$l_\epsilon(f(x), y) = \begin{cases} 0 & \text{if } |f(x) - y| < \epsilon \\ |f(x) - y| - \epsilon & \text{otherwise.} \end{cases} \tag{4}$$

Then we get the following optimization problem:

$$w^\star = \operatorname*{argmin}_{w} C \sum_{i=1}^n l_\epsilon(w^T x_i, y_i) + \frac{1}{2}\|w\|_2^2. \tag{5}$$

1. Write down the dual problem of SVR. (Hint: follow the derivations for SVM)

   Answer:

   We introduce slack variables $\xi_i$, $\xi_i^*$ and write the original problem as

$$\begin{array}{ll} \min & \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n(\xi_i + \xi_i^*) \\ s.t. & \xi_i \geq y_i - w^T x_i - \epsilon \\ & \xi_i^* \geq w^T x_i - y_i - \epsilon \\ & \xi_i, \xi_i^* \geq 0 \end{array} \tag{6}$$

   By introducing slack variables $\eta_i$, $\eta_i^*$, $\alpha_i$, $\alpha_i^*$, the Lagrangian is

$$\begin{aligned} L = \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^n(\xi_i + \xi_i^*) - \sum_{i=1}^n \eta_i \xi_i - \sum_{i=1}^n \eta_i^* \xi_i^* \\ + \sum_{i=1}^n \alpha_i (y_i - w^T x_i - \epsilon - \xi_i) + \sum_{i=1}^n \alpha_i^* (w^T x_i - y_i - \epsilon - \xi_i^*) \end{aligned} \tag{7}$$

   The primal problem is

$$\begin{array}{rl} \displaystyle\min_{w,\xi_i,\xi_i^*} \max_{\alpha_i,\alpha_i^*,\eta_i,\eta_i^*} & L \\ s.t. & \alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0 \end{array} \tag{8}$$

   So the dual problem is

$$\begin{array}{rl} \displaystyle\max_{\alpha_i,\alpha_i^*,\eta_i,\eta_i^*} \min_{w,\xi_i,\xi_i^*} & L \\ s.t. & \alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geq 0 \end{array} \tag{9}$$

We can get $\min_{w,\xi_i,\xi_i^*} L$ by taking the derivatives and solve for $w, \xi_i, \xi_i^*$, so the dual problem is

$$\begin{array}{cl} \max_{\alpha_i,\alpha_i^*,\eta_i,\eta_i^*} & -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)x_i^Tx_j+\sum_{i=1}^n(\alpha_i-\alpha_i^*)y_i-\sum_{i=1}^n(\alpha_i+\alpha_i^*)\epsilon \\ s.t. & 0\le\alpha_i\le C \\ & 0\le\alpha_i^*\le C \end{array} \tag{10}$$

2. Write down the KKT conditions, and explain what are the "support vectors".

   Answer: The KKT conditions are:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^n(\alpha_i-\alpha_i^*)x_i = 0 \tag{11}$$
$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \eta_i = 0 \tag{12}$$
$$\frac{\partial L}{\partial \xi_i^*} = C - \alpha_i^* - \eta_i^* = 0 \tag{13}$$
$$\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \ge 0 \tag{14}$$
$$\eta_i\xi_i = 0, \eta_i^*\xi_i^* = 0 \tag{15}$$
$$\alpha_i(y_i - w^Tx_i - \epsilon - \xi_i) = 0 \tag{16}$$
$$\alpha_i^*(w^Tx_i - y_i - \epsilon - \xi_i^*) = 0 \tag{17}$$

   The support vectors are those with $\alpha_i - \alpha_i^* \neq 0$.

3. Derive a kernelized version of SVR. For a test point $x$, write down the prediction rule.

   Answer: In a kernelized version, for any kernel function $K(\cdot,\cdot)$, just replace the $x_i^Tx_j$ with $K(x_i,x_j)$. For a test point $x$, the prediction rule is

$$f(x) = \sum_{i=1}^n(\alpha_i-\alpha_i^*)K(x_i,x) \tag{18}$$

4. Give one reason why do we usually solve the dual problem of SVR and SVM instead of the primal.

   Answer: We can introduce kernel functions here.

5. Implement SVR on a 1-D toy dataset. Each line of the dataset contains a training instance $(x_i, y_i)$ (separated by a tab). For this problem, you need to

   - Use RBF kernel $k(x_i,x_j) = \exp(-\frac{\|x_i-x_j\|_2^2}{2h^2})$, and take $h = 0.5$, $C = 4$, $\epsilon = 0.1$.
   - Plot the prediction curve for $x \in [0,1]$ and show the support vectors versus other training points in the training dataset.

   (Hint: You are allowed to use optimization toolkits such as CVX or Matlab's inbuilt function quadprog to solve the dual problem.)

   Answer:

```
clc;
clear;

data = dlmread('SVR_dataset.txt');
```

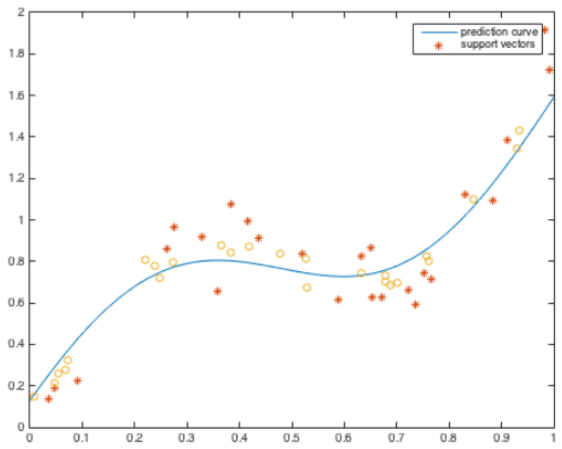

Figure 3: SVR results.

```
x = data(:, 1);
y = data(:, 2);

h = 0.5;
C = 4;
eps = 0.1;

n = size(x, 1);

% Parameters are 2n by 1 vector, whose first n elements are
% p = alpha - alpha^star and next n elements are
% q = alpha + alpha^star

H = zeros(2 * n, 2 * n);
for i = 1:n
    for j = 1:n
        temp = norm(x(i) - x(j), 2);
        H(i, j) = exp(- temp ^ 2 / 2 / h ^ 2);
    end
end

f = - eps * ones(2 * n, 1);
f(1:50, 1) = y;

A = zeros(4 * n, 2 * n);
for i = 1:n
    A(i, i) = 0.5;
    A(i, i + n) = 0.5;

    A(i + n, i) = -0.5;
    A(i + n, i + n) = 0.5;

    A(i + 2 * n, i) = -0.5;
```

```
        A(i + 2 * n, i + n) = -0.5;

        A(i + 3 * n, i) = 0.5;
        A(i + 3 * n, i + n) = -0.5;
    end

    b = zeros(4 * n, 1);
    b(1:2 * n, 1) = C;

    params = quadprog(H, -f, A, b);

    %% Plot

    p = params(1:n, 1);
    q = params(n + 1:100, 1);

    limit = 0.00001;
    sv_idx = find(abs(p) > limit);
    nsv_idx = find(abs(p) <= limit);

    x_sv = x(sv_idx, 1);
    x_nsv = x(nsv_idx, 1);
    y_sv = y(sv_idx, 1);
    y_nsv = y(nsv_idx, 1);

    cut = 99;
    x_axis = 0:1/cut:1;
    y_axis = zeros(size(x_axis));

    for i = 1:cut + 1
        K = zeros(n, 1);
        for j = 1:n
            K(j, 1) = exp( - norm(x_axis(i) - x(j), 2) ^ 2 / 2 / h ^ 2);
        end
        y_axis(i) = dot(K, p);
    end

    figure % opens new figure window
    plot(x_axis, y_axis);
    hold on;
    scatter(x_sv, y_sv, '*');
    hold on;
    scatter(x_nsv, y_nsv);
    legend('prediction curve','support vectors');
```

### 2.2 Support Kernel Machines (20 Points)

In SVM, the kernel function can be viewed as a similarity measure between data points. In some classification scenarios, features may come from different sources or modalities, e.g. in some tasks the data may contain both image features and text features. In that case, since these are different representations, they have different measures of similarity corresponding to different kernels. In such a case, we want to learn a combination of kernels instead of using a single kernel. There is significant amount of work in combing kernels, here we adapt the notations in [1].

We begin by considering a linear case of Support Kernel Machine (SKM). Suppose the data points $x_i \in \mathcal{X} = \mathbb{R}^k$. We also assume we are given a decomposition of $\mathbb{R}^k = \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_m}$, so that each data point $x_i$ can be decomposed into $m$ block components, i.e. $x_i = (x_{1i}, \cdots, x_{mi})$ where each $x_{ji}$ is in general a vector. In real tasks, each block may correspond to a certain kind of representation, e.g. $x_{1i}$ may correspond to image features and $x_{2i}$ may be text features.

Our goal is to find a linear classifier of the form $y = \text{sign}(w^T x + b)$ where $w$ has the same block decomposition $w = (w_1, \cdots, w_m) \in \mathbb{R}^{k_1 + \cdots + k_m}$. Recall that in linear SVM the objective is:

$$\underset{\substack{w \in \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_m} \\ \xi_i \geq 0,\ b \in \mathbb{R}}}{\text{minimize}} \quad \frac{1}{2}\|w\|_2^2 + C\sum_{i=1}^{n} \xi_i \tag{19}$$

$$\text{subject to} \quad y_i(\sum_j w_j^T x_{ji} + b) \geq 1 - \xi_i,\ \forall i \in \{1, \cdots, n\} \tag{20}$$

In SKM, we encourage the sparsity of the vector $w$ at the level of blocks. The primal problem for the SKM is defined as:

$$\underset{\substack{w \in \mathbb{R}^{k_1} \times \cdots \times \mathbb{R}^{k_m} \\ \xi_i \geq 0,\ b \in \mathbb{R}}}{\text{minimize}} \quad \frac{1}{2}(\sum_{j=1}^{m} d_j \|w_j\|_2)^2 + C\sum_{i=1}^{n} \xi_i \tag{21}$$

$$\text{subject to} \quad y_i(\sum_j w_j^T x_{ji} + b) \geq 1 - \xi_i,\ \forall i \in \{1, \cdots, n\} \tag{22}$$

where $d_j > 0$ can be seen as constant.

1. By introducing dual variables $\alpha_i \geq 0$ and $\beta_i \geq 0$, we get the Lagrangian function

$$\mathcal{L} = \frac{1}{2}(\sum_{j=1}^{m} d_j \|w_j\|_2)^2 + C\sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \alpha_i (y_i (\sum_{j=1}^{m} w_j^T x_{ji} + b) - 1 + \xi_i) - \sum_{i=1}^{n} \beta_i \xi_i \tag{23}$$

Denote $\gamma = \sum_{j=1}^{m} d_j \|w_j\|_2$.

(a) Show that at the minimum of the Lagrangian function, i.e. $w = \text{argmin}_w \mathcal{L}$ for this and the following questions,

$$\|w_j\|_2 d_j \gamma = w_j^T \sum_{i=1}^{n} \alpha_i y_i x_{ji},\ \forall j \in \{1, \cdots, m\} \tag{24}$$

Answer: It is apparent that the conclusion holds when $w_j = 0$. When $w_j \neq 0$, we have

$$\frac{\partial L}{\partial w_j} = \gamma d_j \frac{w_j}{\|w_j\|_2} - \sum_{i=1}^{n} \alpha_i y_i x_{ji} \tag{25}$$

At the minimum, $\frac{\partial L}{\partial w_j} = 0$, we have

$$\gamma d_j \frac{w_j}{\|w_j\|_2} = \sum_{i=1}^{n} \alpha_i y_i x_{ji} \tag{26}$$

Multiply both sides by $w_j^T$, we have

$$\|w_j\|_2 \gamma d_j = w_j^T \sum_{i=1}^{n} \alpha_i y_i x_{ji} \tag{27}$$

(b) Show that $\| \sum_{i=1}^{n} \alpha_i y_i x_{ji} \|_2 \leq d_j \gamma$, $\forall j \in \{1, \cdots, m\}$.

Note: for (b) you can get full credit if you only consider $w_j \neq 0$, but you can get 5 extra points if you include $w_j = 0$ case in your proof. A hint is that $\mathcal{L}$ is not differentiable w.r.t. $w_j$ if $w_j = 0$, and you may refer to this link for how to deal with that case by using $\partial \|x\|_2 = \{g : \|g\|_2 \leq 1\}$ if $x = 0$.

Answer: If $w_j \neq 0$. Use the intermediate results of (a), we have

$$\gamma d_j \frac{w_j}{\|w_j\|_2} = \sum_{i=1}^{n} \alpha_i y_i x_{ji} \tag{28}$$

Take the $l_2$ norm of both sides, we have

$$\gamma d_j = \| \sum_{i=1}^{n} \alpha_i y_i x_{ji} \|_2 \tag{29}$$

If $w_j = 0$, $0 \in \partial L$, we have that for some $g$ with $\|g\|_2 \leq 1$,

$$\gamma d_j g = \sum_{i=1}^{n} \alpha_i y_i x_{ji} \tag{30}$$

Take the $l_2$ norm of both sides, we have

$$\gamma d_j \|g\|_2 = \| \sum_{i=1}^{n} \alpha_i y_i x_{ji} \|_2 \tag{31}$$

As $\|g\|_2 \leq 1$, we have $d_j \gamma \geq \| \sum_{i=1}^{n} \alpha_i y_i x_{ji} \|_2$.

(c) Show that

- if $\| \sum_i \alpha_i y_i x_{ji} \|_2 < d_j \gamma$, then $w_j = 0$,
- if $\| \sum_i \alpha_i y_i x_{ji} \|_2 = d_j \gamma$, then $\exists \eta_j \geq 0$, such that $w_j = \eta_j \sum_i \alpha_i y_i x_{ji}$.

Answer: If $w_j \neq 0$, using the intermediate results of (b) we have

$$\gamma d_j = \| \sum_{i=1}^{n} \alpha_i y_i x_{ji} \|_2 \tag{32}$$

So if $\| \sum_{i=1}^{n} \alpha_i y_i x_{ji} \|_2 < d_j \gamma$, $w_j = 0$.

On the other hand, we have proved in (a) that if $w_j \neq 0$, $\gamma d_j \frac{w_j}{\|w_j\|_2} = \sum_{i=1}^{n} \alpha_i y_i x_{ji}$. Let $\eta_j$ be $\frac{\|w_j\|_2}{\gamma d_j}$, then $w_j = \eta_j \sum_{i=1}^{n} \alpha_i y_i x_{ji}$.

2. Recall from homework 2 that $\ell_1$ norm can encourage sparsity. Explain the effect of the regularization term $\frac{1}{2}(\sum_{j=1}^{m} d_j \|w_j\|_2)^2$.

Answer: Recall that $l_1$ regularizer has the effect of putting some coefficients to zero, and if $d_j \|w_j\|_2 = 0$, then $w_j = 0$. Therefore some blocks $w_j$ tend to zero due to this regularizer (group lasso).

3. Now we extend the above analysis to a kernelized version. Assume that we have a mapping $\phi : \mathcal{X} \to \mathbb{R}^f$ which is generally a non-linear function. We assume that $\phi(x)$ has $m$ block components $\phi(x) = (\phi_1(x), \cdots, \phi_m(x))$red, and we also assume $w$ has the same decomposition $w = (w_1, \cdots, w_m)$. Show that at the minimum of the Lagrangian function, $\exists \eta_j \geq 0$ such that $w_j = \eta_j \sum_{i=1}^{n} \alpha_i y_i \phi_j(x_i)$.

Answer: Since it is a mapping for $x$, we can simply substitute $x_{ji}$ with $\phi_j(x_i)$ and every conclusions above still hold. The proof completes by using the conclusion in 1(c) and replace $x_{ji}$ with $\phi_j(x_i)$.

### 2.3 SVM Error Analysis (5 Points)

In this problem, we want to analyze the error of SVM classification. Assume that we have $n$ data points $(x_i, y_i)_{i=1}^n$ where $x_i \in \mathbb{R}^m$ and $y_i = 1, \cdots, K$. Assume that we train an SVM classifier $f_{(x_1,y_1),\cdots(x_n,y_n)}$ on these $n$ data points.

For a randomly drawn test data point $(x_{n+1}, y_{n+1})$, the prediction is $y_{n+1}^{\text{pred}} = f_{(x_1,y_1),\cdots,(x_n,y_n)}(x_{n+1})$. We assume that the $n$ training data points and the test data point $(x_{n+1}, y_{n+1})$ are drawn i.i.d from some unknown underlying distribution. The expected error rate is defined as:

$$\mathsf{err} = \mathbb{E}_{(x_1,y_1),\cdots(x_n,y_n)}\mathbb{E}_{(x_{n+1},y_{n+1})}[\mathbf{1}\left\{f_{(x_1,y_1),\cdots,(x_n,y_n)}(x_{n+1}) \neq y_{n+1}\right\}] \tag{33}$$

where the indicator function $\mathbf{1}\{A\} = 1$ if $A$ is true, otherwise 0.

1. Show the the expected error rate is equal to the expectation of leave-one-out cross validation error for $n+1$ data points.

   Answer: As the points are i.i.d, we have for any $i$,

$$\begin{aligned}&\mathbb{E}_{(x_1,y_1),\cdots(x_n,y_n)}\mathbb{E}_{(x_{n+1},y_{n+1})}[\mathbf{1}\left\{f_{(x_1,y_1),\cdots,(x_n,y_n)}(x_{n+1}) \neq y_{n+1}\right\}]\\ &\qquad= \mathbb{E}_{(x_1,y_1),\cdots,(x_{n+1},y_{n+1})}[\mathbf{1}\left\{f_{(x_1,y_1),\cdots,(x_{i-1},y_{i-1}),(x_{i+1},y_{i+1}),\cdots,(x_{n+1},y_{n+1})}(x_i) \neq y_i\right\}]\end{aligned} \tag{34}$$

   By looping through $i = 1$ to $i = n+1$ and averaging, we have

$$\begin{aligned}&\mathbb{E}_{(x_1,y_1),\cdots(x_n,y_n)}\mathbb{E}_{(x_{n+1},y_{n+1})}[\mathbf{1}\left\{f_{(x_1,y_1),\cdots,(x_n,y_n)}(x_{n+1}) \neq y_{n+1}\right\}]\\ &= \frac{1}{n+1}\sum_{i=1}^{n+1}\mathbb{E}_{(x_1,y_1),\cdots,(x_{n+1},y_{n+1})}[\mathbf{1}\left\{f_{(x_1,y_1),\cdots,(x_{i-1},y_{i-1}),(x_{i+1},y_{i+1}),\cdots,(x_{n+1},y_{n+1})}(x_i) \neq y_i\right\}]\\ &= \mathbb{E}_{(x_1,y_1),\cdots,(x_{n+1},y_{n+1})}[\frac{1}{n+1}\sum_{i=1}^{n+1}\mathbf{1}\left\{f_{(x_1,y_1),\cdots,(x_{i-1},y_{i-1}),(x_{i+1},y_{i+1}),\cdots,(x_{n+1},y_{n+1})}(x_i) \neq y_i\right\}]\end{aligned} \tag{35}$$

   The right hand side is the expectation of the leave on out error rate.

2. In the lecture, we have the statement that "the leave-one-out cross-validation error does not depend on the dimensionality of the feature space but only on the number of support vectors". Show that this statement is true by explaining why

$$\mathsf{err}_{\text{loocv}} \leq \frac{n_s}{n+1} \tag{36}$$

   where $\mathsf{err}_{\text{loocv}}$ is the leave-one-out cross-validation error for training set $(x_i, y_i)_{i=1}^{n+1}$, $n_s$ is the number of support vectors.

   Answer: If we remove a non-support vector and retrain the SVM, the decision boundary before removing it is still the optimum, and if we use that decision boundary, the previous non-support vector will not be misclassified. So the leave one out error possibly occurs when we remove the support vectors, hence

$$\mathsf{err}_{\text{loocv}} \leq \frac{n_s}{n+1} \tag{37}$$

## Acknowlegements

Thanks for Desai's homework for providing this solution.

## References

[1] Francis R Bach, Gert R.G. Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004. 10