AI534 — Written Homework Assignment 2 (45 pts) —

This assignment covers Kernel methods and Support vector machines.

1. (Cubic Kernels.) (8 pts) In class, we showed that the quadratic kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^2$ was equivalent to mapping each $\mathbf{x} = (x_1, x_2) \in R^2$ into a higher dimensional space where

$$\Phi(\mathbf{x}) = (x_1^2, x_2^2, \sqrt{2}x_1x_2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

Now consider the cubic kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^3$. What is the corresponding Φ function?

We use $x = (x_1, x_2)$ to calculate the cubic kernel $K(x_i, x_j) = (x_i \cdot x_j + 1)^3$. For example, we can apply $x_i = (x_{i1}, x_{i2})$ and $x_j = (x_{j1}, x_{j2})$ to the cubic kernel.

$$(x_i \cdot x_j + 1)^3 = (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^3$$

$$= x_{i1}^3x_{j1}^3 + x_{i2}^3x_{j2}^3 + 3x_{i1}x_{j1}x_{i2}^2x_{j2}^2 + 3x_{i1}^2x_{j1}^2x_{i2}x_{j2} + 3x_{i1}^2x_{j1}^2$$

$$+3x_{i2}^2x_{j2}^2 + 6x_{i1}x_{j1}x_{i2}x_{j2} + 3x_{i1}x_{j1} + 3x_{i2}x_{j2} + 1$$

From the calculation above, we can guess the corresponding Φ function like:

$$\Phi(x_i) \cdot \Phi(x_j) = (x_{i1}^3, x_{i2}^3, \sqrt{3}x_{i1}x_{i2}^2, \sqrt{3}x_{i1}^2x_{i2}, \sqrt{3}x_{i1}^2, \sqrt{3}x_{i2}^2, \sqrt{6}x_{i1}x_{i2}, \sqrt{3}x_{i1}, \sqrt{3}x_{i2}, 1)$$
$$\cdot(x_{j1}^3, x_{j2}^3, \sqrt{3}x_{j1}x_{j2}^2, \sqrt{3}x_{j1}^2x_{j2}, \sqrt{3}x_{j1}^2, \sqrt{3}x_{j2}^2, \sqrt{6}x_{j1}x_{j2}, \sqrt{3}x_{j1}, \sqrt{3}x_{j2}, 1)$$

Since the product of two Φ functions should hold the cubic kernel calculation above. Thus, we finally get:

$$\Phi(\mathbf{x}) = (x_1^3, x_2^3, \sqrt{3}x_1^2x_2, \sqrt{3}x_1x_2^2, \sqrt{3}x_1^2, \sqrt{3}x_2^2, \sqrt{3}x_1, \sqrt{3}x_2, \sqrt{6}x_1x_2, 1)$$

2. (Kernel or not). (5 pts) Suppose that K_1 and K_2 are kernels with feature maps ϕ_1 and ϕ_2 respectively. Is function $K(\mathbf{x}, \mathbf{z}) = c_1K_1(\mathbf{x}, \mathbf{z}) + c_2K_2(\mathbf{x}, \mathbf{z})$ for $c_1, c_2 > 0$ a kernel function? If your answer is yes, write down the corresponding ϕ in terms of ϕ_1 and ϕ_2. If not, provide a proof or explain why.

The function $K(x, z) = c_1K_1(x, z) + c_2K_2(x, z)$ for $c_1, c_2 > 0$ is a kernel function. According to the closure property of kernels in the lecture slides, it can be used to prove whether the fucntion above is a kernel function or not. It describes that If K_1 and K_2 are kernel functions, then the following are all kernel functions:

$$K(x, y) = K_1(x, y) + K_2(x, y) \cdots (1)$$
$$\Phi = \textit{concatenation of } \Phi_1 \textit{ and } \Phi_2$$

$$K(x, y) = aK_1(x, y), \textit{ where } a > 0 \cdots (2)$$
$$\Phi = \sqrt{a}\Phi_1$$

Firstly, we utilize (2) for the function $K(x, z) = c_1K_1(x, z) + c_2K_2(x, z)$ for $c_1, c_2 > 0$, meaning that:

$$K(x, z) = c_1K_1(x, z), \textit{ where } c_1 > 0$$

$$K(x, z) = c_2K_2(x, z), \textit{ where } c_2 > 0$$

Then, we apply (1) to the two equations above:

$$K(x,z) = c_1 K_1(x,z) + c_2 K_2(x,z), \text{ where } c_1, c_2 > 0$$

Thus, it is a kernel function.

3. Kernelizing Logistic Regression (10 pts) For this problem you will follow the example of kernelizing perceptron, to kernelize the logistic regression shown below.

Algorithm 1: Stochastic gradient descent for logistic regression

Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$(training data), γ(learning rate)
Output: learned weight vector $\mathbf{w}$

```
Initialize w = 0;
while not converged do
    for i = 1, ..., N do
        w ← w + γ(y_i − σ(w^T x_i))x_i
    end
end
```

Specifically, please:

(a) (4 pts) Argue that the solution $\mathbf{w}^*$ for logistic regression can be expressed as the weighted sum of training examples (similar to slide 8 of the kernel methods lecture)

Based on the Stochastic gradient descent for logistic regression in this assignment, the weight w can be expressed as the weighted sum of training examples since the method updating w is dependent on x_i in the for-loop $(i, ..., N)$. So, we can express it as:

$$w \leftarrow w + \gamma \sum_{i=1}^{N} (y_i - \sigma(w^T x_i)) x_i$$

The update above can be simplified with $\alpha_i = \gamma(y_i - \sigma(w^T x_i))$. So, we get:

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i x_i$$

Thus, the solution $\mathbf{w}^*$ for logistic regression can be expressed as the weighted sum of training examples.

(b) (6 pts) Modify the following stochastic gradient descent algorithm logistic regression algorithm to kernelize it. (Hint: similar to the bottom algorithm on slide 14 of the kernel method lecture, but instead of counter, you will learn a continuous weights for α's)

As derived in part (a), w can be represented as a weighted sum of training examples:

$$\mathbf{w}^* = \sum_{i=1}^{N} \alpha_i x_i$$

, where α_i represents the weights for each training example.
$\hat{y}$ is equivalent to the term $w^T x$. We can apply the weight w^* to $\hat{y}$, meaning that:

$$\hat{y} = (\sum_{i=1}^{N} \alpha_i x_i)^T x$$
$$= \sum_{i=1}^{N} \alpha_i x_i^T x$$
$$= \sum_{i=1}^{N} \alpha_i K(x_i, x)$$

Then, $\hat{y}$ can be expressed as:

$$\hat{y}_i = \sum_{i=1}^{N} \alpha_i K(x_i, x)$$

The update rule of the algorithm is:

$$\alpha_i \leftarrow \alpha_i + \gamma(y_i - \sigma(\sum_{j=1}^{N} \alpha_i K(x_j, x)))$$

Thus, the algorithm is below.

Algorithm 2: Kernelized Stochastic gradient descent for logistic regression

Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$*(training data),* γ*(learning rate)*
Output: *learned weight vector* $\mathbf{w}$

```
Initialize w = 0;
while not converged do
    for i = 1, ..., N do
        ŷ_i ← Σ_{i=1}^N α_i K(x_i, x)
        α_i ← α_i + γ(y_i − σ(Σ_{j=1}^N α_i K(x_j, x)))
    end
end
```

4. (Hard margin SVM) (8 pts) Apply linear SVM without soft margin to the following problem.

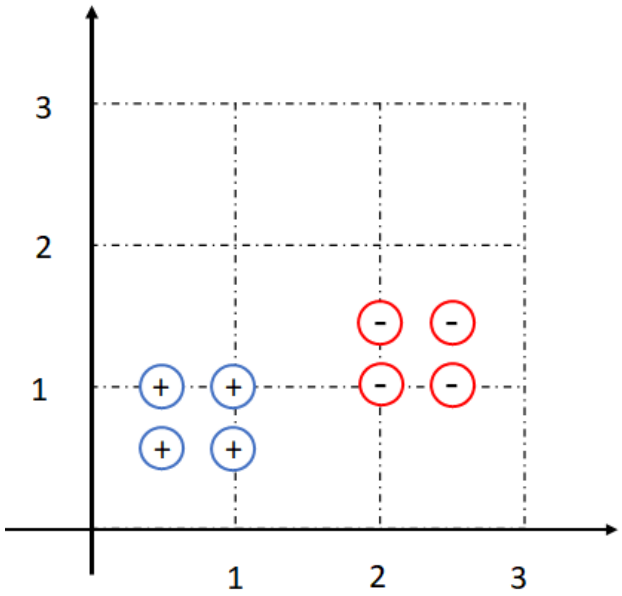

Figure 1:

a. (3pts) Please mark out the support vectors, the decision boundary ($w_1x_1 + w_2x_2 + b = 0$) and $w_1x_1 + w_2x_2 + b = 1$ and $w_1x_1 + w_2x_2 + b = -1$. You don't need to solve the optimization problem for this, you should be able to eyeball the solution and find the linear separator with the largest margin.

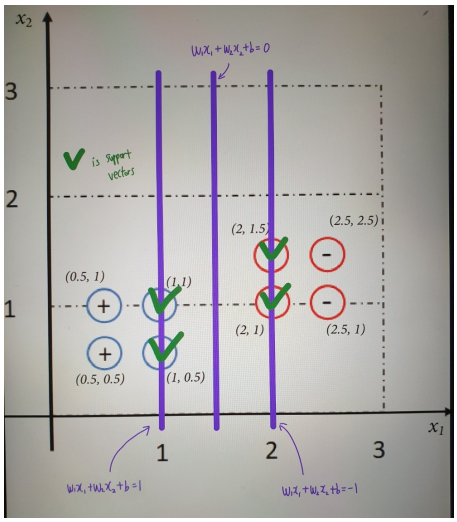

Figure 2: Support vectors and Decision Boundaries

As can be seen in Figure 2., green checks are support vectors, and purple lines are decision boundaries where the left is $w_1x_1 + w_2x_2 + b = 1$, the middle is $w_1x_1 + w_2x_2 + b = 0$, and the right is $w_1x_1 + w_2x_2 + b = -1$.

b. (5 pts) Please solve for w_1, w_2 and b based on the support vectors you identified in (a). Hint: the support vectors would have functional margin = 1.

We use positive and negative class support vector decision boundary to solve this problem. These are:

$$w_1x_1 + w_2x_2 + b = 1 \text{ (positive support vector decision boundary)}$$

$$w_1x_1 + w_2x_2 + b = -1 \text{ (negative support vector decision boundary)}$$

The positive support vectors are (1, 1) and (1, 0.5). Otherwise, the negative support vectors are (2, 1) and (2, 1.5). We apply these support vectors to each decision boundaries:

$$w_1 \cdot 1 + w_2 \cdot 1 + b = 1 \;\Rightarrow\; w_1 + w_2 + b = 1 \cdots (1)$$

$$w_1 \cdot 2 + w_2 \cdot 1 + b = 1 \;\Rightarrow\; 2w_1 + w_2 + b = -1 \cdots (2)$$

Then, we can get w_1 and w_2 by subtracting (1) and (2). So, the results are:

$$w_1 = -2,\; w_2 = 3 - b$$

In the hint, the support vectors would have functional margin = 1, meaning that:

$$-2 \cdot x_1 + (3 - b) \cdot x_2 + b = 1$$

Next, we apply other support vector in the positive support vector decision boundary, which is (1, 0.5).

$$-2 \cdot 1 + (3 - b) \cdot 0.5 + b = 1$$

$$\Rightarrow -2 + 1.5 - 0.5b + b = 1$$

$$\Rightarrow 0.5b = 1.5$$

$$\Rightarrow b = 3$$

Thus, $w_1 = -2$, $w_2 = 0$, and $b = 3$

5. L_2 SVM (14 pts)
Given a set of training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where $y_i \in \{1, -1\}$ for all i. The following is the primal formulation of L_2 SVM, a variant of the standard SVM obtained by squaring the slacks.

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \; & \frac{1}{2}\mathbf{w}^T\mathbf{w} + c\sum_{i=1}^{N}\xi_i^2 \\ \text{s.t. } & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i,\; i \in \{1, \cdots, N\} \\ & \xi_i \geq 0,\; i \in \{1, \cdots, N\} \end{aligned}$$

a. (3pts) Show that removing the second constraint $\xi_i \geq 0$ will not change the solution to the problem. In other words, let $(\mathbf{w}^*, b^*, \xi^*)$ be the optimal solution to the problem without this set of constraints, show that $\xi_i^* \geq 0$ must be true, $\forall i \in \{1, \cdots, N\}$. (Hint: use proof by contradiction by assuming that there exists some $\xi_i^* < 0$.)

Let's use contradiction for proof. Assume that there exists some optimal solution $(\mathbf{w}^*, b^*, \xi^*)$ when $\xi_i^* < 0$.

If we set $\xi_i^* < 0$, the L_2 SVM would be:

$$\min_{\mathbf{w},b,\xi} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + c\sum_{i=1}^{N}\xi_i^2, \ \textit{where } c\sum_{i=1}^{N}\xi_i^2 > 0$$

If we set $\xi_i^* = 0$, the L_2 SVM would be:

$$\min_{\mathbf{w},b,\xi} \ \frac{1}{2}\mathbf{w}^T\mathbf{w}$$

From these, the term $c\sum_{i=1}^{N}\xi_i^2$ is always positive and cannot be smaller than when if $\xi_i^* = 0$, meaning that there is no impact of ξ_i^* on the function.
Thus, $\xi_i^* >= 0$ must be true, so $(\mathbf{w}, b, \xi)$ is the be optimal solution to the problem without this set of constraints.

b. (3 pts) After removing the second set of constraints, we have a simpler problem with only one set of constraints. Now provide the lagrangian of this new problem.

We have a simpler problem with only one set of constraints, which is:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \ i \in \{1, \cdots, N\}$$

This constraint can be modified as:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i \geq 0$$

We can apply it to the Lagrangian of this new problem, which is:

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + c\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i(y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \xi_i), \ \textit{where } \alpha_i >= 0$$

c. (8pts) Derive the dual of this problem. How is it different from the standard SVM with hinge loss? Which formulation is more sensitive to outliers?

According to the "Aside: Constrained Optimization" in the lecture slides, solving the dual problem is that $\text{argmax}_{\alpha>=0}\,\text{argmin}_{w,b}\,\mathcal{L}(\mathbf{w}, b, \alpha)$.
Take the gradient of $\mathcal{L}$ w.r.t. $\mathbf{w}$ and b and set to zero:
The case of $\mathbf{w}$:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i$$

$$\mathbf{w} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i = 0$$

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

The case of b:

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0$$

$$\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0$$

The case of ξ_i:

$$\frac{\partial \mathcal{L}}{\partial \xi_i} = 2c\xi_i - \alpha_i = 0$$

$$\xi_i = \frac{\alpha_i}{2c}$$

Substitute $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ and $\xi_i = \frac{\alpha_i}{2c}$ into the Lagrangian Dual problem:

$$\underset{\alpha}{\text{argmax}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \frac{1}{4c} \sum_{i=1}^{N} \alpha_i^2$$

$$, s.t. \sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \alpha_i >= 0$$

L_2 SVM uses a quadratic penalty $\sum_{i=1}^{N} \xi_i$, making it more resistant to large deviations in outliers. On the other hand, standard SVM with hinge loss uses a linear penalty, making it more sensitive to outliers.

Thus, L_2 SVM is generally less sensitive to outliers due to the quadratic penalty whereas standard SVM with hinge loss is more sensitive to outliers since large slack values can significantly impact the function.