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?

Let $\mathbf{x}_i = (x_{i1}, x_{i2})$ and $\mathbf{x}_j = (x_{j1}, x_{j2})$. Then the kernel is

$$\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^3 \\
&= (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^2 \cdot (x_{i1}x_{j1} + x_{i2}x_{j2} + 1) \\
&= (x_{i1}^2x_{j1}^2 + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 1) \cdot (x_{i1}x_{j1} + x_{i2}x_{j2} + 1) \\
&= x_{i1}^3x_{j1}^3 + 3x_{i1}^2x_{j1}^2 + 3x_{i1}x_{j1} + \\
&\quad 3x_{i1}^2x_{j1}^2x_{i2}x_{j2} + 6x_{i1}x_{j1}x_{i2}x_{j2} + 3x_{i1}x_{j1}x_{i2}^2x_{j2}^2 + \\
&\quad 3x_{i2}x_{j2} + 3x_{i2}^2x_{j2}^2 + x_{i2}^3x_{j2}^3 + 1 \\
&= (x_{i1}^3, \sqrt{3}x_{i1}^2, \sqrt{3}x_{i1}, \sqrt{3}x_{i1}^2x_{i2}, \sqrt{6}x_{i1}x_{i2}, \sqrt{3}x_{i1}x_{i2}^2, \sqrt{3}x_{i2}, \sqrt{3}x_{i2}^2, x_{i2}^3, 1) \cdot \\
&\quad (x_{j1}^3, \sqrt{3}x_{j1}^2, \sqrt{3}x_{j1}, \sqrt{3}x_{j1}^2x_{j2}, \sqrt{6}x_{j1}x_{j2}, \sqrt{3}x_{j1}x_{j2}^2, \sqrt{3}x_{j2}, \sqrt{3}x_{j2}^2, x_{j2}^3, 1)
\end{aligned}$$

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

2. (Kernel or not). (10 pts) In the following problems, suppose that K, K_1 and K_2 are kernels with feature maps ϕ, ϕ_1 and ϕ_2. For the following functions $K'(x, z)$, state if they are kernels or not. If they are kernels, write down the corresponding ϕ in terms of ϕ, ϕ_1 and ϕ_2. If they are not kernels, prove that they are not.

   - (2 pts) $K'(\mathbf{x}, \mathbf{z}) = cK(\mathbf{x}, \mathbf{z})$ for $c > 0$.
     $K'(\mathbf{x}, \mathbf{z}) = cK(\mathbf{x}, \mathbf{z}) = c\langle\phi(\mathbf{x}), \phi(\mathbf{z})\rangle = \langle\sqrt{c}\phi(\mathbf{x}), \sqrt{c}\phi(\mathbf{z})\rangle$
   - (2 pts) $K'(\mathbf{x}, \mathbf{z}) = cK(\mathbf{x}, \mathbf{z})$ for $c < 0$.
     It is not a kernel. Since $c < 0$, we know $-cK(\mathbf{x}, \mathbf{z})$ is a kernel function and its gram matrix for any given training data will be positive semi-definite (with non negative eigenvalues). The gram matrix for $cK(\mathbf{x}, \mathbf{z})$ will be the negative of that matrix, which will not be positive semi-definite (assuming K does not always return zero), violating the condition for being a kernel function.
   - (2 pts) $K'(\mathbf{x}, \mathbf{z}) = c_1K_1(\mathbf{x}, \mathbf{z}) + c_2K_2(\mathbf{x}, \mathbf{z})$ for $c_1, c_2 > 0$.
     $K'(\mathbf{x}, \mathbf{z}) = c_1K_1(\mathbf{x}, \mathbf{z}) + c_2K_2(\mathbf{x}, \mathbf{z}) = c_1\langle\phi_1(\mathbf{x}), \phi_1(\mathbf{z})\rangle + c_2\langle\phi_2(\mathbf{x}), \phi_2(\mathbf{z})\rangle = \langle\sqrt{c_1}\phi_1(\mathbf{x}), \sqrt{c_1}\phi_1(\mathbf{z})\rangle + \langle\sqrt{c_2}\phi_2(\mathbf{x}), \sqrt{c_2}\phi_2(\mathbf{z})\rangle = \langle[\sqrt{c_1}\phi_1(\mathbf{x}), \sqrt{c_2}\phi_2(\mathbf{x})], [\sqrt{c_2}\phi_1(\mathbf{z}), \sqrt{c_2}\phi_2(\mathbf{z})]\rangle$
     So $\phi'(\mathbf{x})$ is the concatenation of $\sqrt{c_1}\phi_1(\mathbf{x})$ and $\sqrt{c_2}\phi_2(\mathbf{x})$.
   - (4 pts) $K'(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$.

$$\begin{aligned}
K'(\mathbf{x}, \mathbf{z}) &= K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z}) \\
&= \langle\phi_1(\mathbf{x}), \phi_1(\mathbf{z})\rangle\langle\phi_2(\mathbf{x}), \phi_2(\mathbf{z})\rangle \\
&= \left(\sum_i \phi_{1,i}(\mathbf{x})\phi_{1,i}(\mathbf{z})\right)\left(\sum_j \phi_{2,j}(\mathbf{x})\phi_{2,j}(\mathbf{z})\right) \\
&= \sum_i \sum_j \left(\phi_{1,i}(\mathbf{x})\phi_{1,i}(\mathbf{z})\right)\left(\phi_{2,j}(\mathbf{x})\phi_{2,j}(\mathbf{z})\right) \\
&= \sum_i \sum_j \left(\phi_{1,i}(\mathbf{x})\phi_{2,j}(\mathbf{x})\right)\left(\phi_{1,i}(\mathbf{z})\phi_{2,j}(\mathbf{z})\right)
\end{aligned}$$

So the new mapping is: $\phi'(\mathbf{x}) = \begin{bmatrix} \phi_{1,1}(\mathbf{x})\phi_{2,1}(\mathbf{x}) \\ \phi_{1,1}(\mathbf{x})\phi_{2,2}(\mathbf{x}) \\ \dots \\ \phi_{1,1}(\mathbf{x})\phi_{2,d_2}(\mathbf{x}) \\ \phi_{1,2}(\mathbf{x})\phi_{2,1}(\mathbf{x}) \\ \phi_{1,2}(\mathbf{x})\phi_{2,2}(\mathbf{x}) \\ \dots \\ \phi_{1,d_1}(\mathbf{x})\phi_{2,d_2}(\mathbf{x}) \end{bmatrix}$

3. Kernelizing Logistic Regression (7pts) For tihs 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 $\mathbf{w}$

1 Initialize $\mathbf{w} = \mathbf{0}$;
2 **while** *not converged* **do**
3 **for** $i = 1, ..., N$ **do**
4 $\mathbf{w} \leftarrow \mathbf{w} + \gamma(y_i - \sigma(\mathbf{w}^T\mathbf{x}_i))\mathbf{x}_i$
5 **end**
6 **end**

Specifically, please:

(a) (2pts) 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)
This is straightforward, we only need to note that line 4 is simply adding a (weighted) training example to $\mathbf{w}$*. Hence, the resulting* $\mathbf{w}$ *will always be a weighted sum of the training examples.*

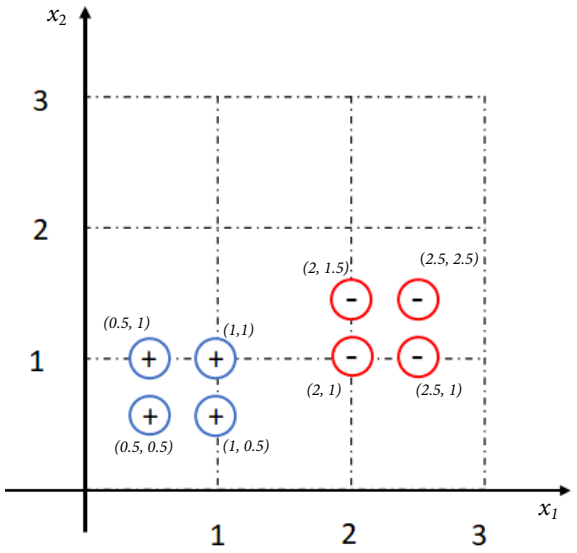
(b) (5 pts) Modify the following stochastic gradient descent algorithm logistic regression algorithm to kernelize it. (Hint: similar to the bottom algorithm on slide 14, but instead of counter, you will learn a continuous weights for α's)

Algorithm 2: Kernelized Stochastic Gradient Descent for Logistic Regression

Input: $\{(\mathbf{x}_i, y_i)_{i=1}^N\}$*(training data),* γ*(learning rate),* $K(\cdot,\cdot)$*(kernel function)*
Output: *learned weight coefficients* α

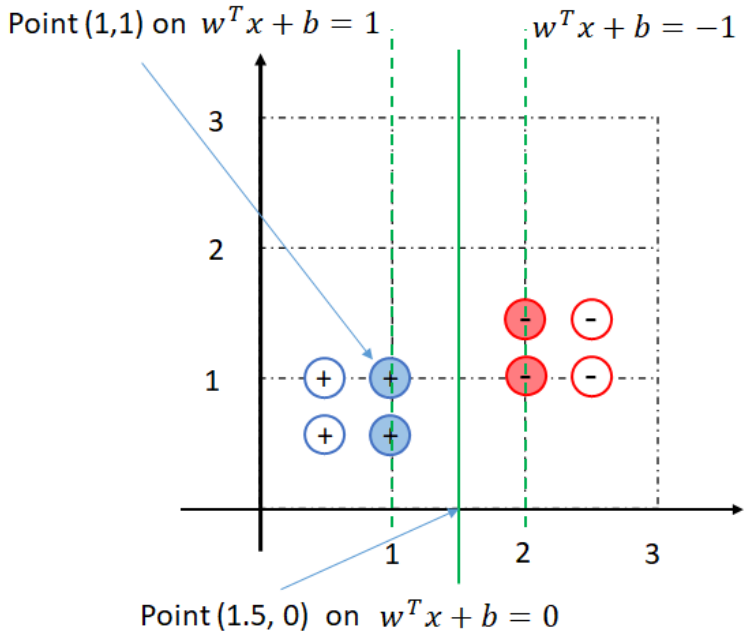
1 *Initialize* $\alpha = \mathbf{0}$*;*
2 **while** *not converged* **do**
3 **for** $i = 1, ..., N$ **do**
4 $z \leftarrow \sum_{j=1}^N \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j)$*;*
5 $\alpha_i \leftarrow \alpha_i + \gamma(y_i - \sigma(z))$
6 **end**
7 **end**

4. (Hard margin SVM) (6 pts) Apply linear SVM without soft margin to the following problem.



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.

See the figure. Filled points are the support vectors.



b. (3 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. *Note that we can tell the equation for the decision boundary (being vertical) must have $w_2 = 0$.*
Next, we know that the Support Vector $(1, 1)$ lies on $w_1 x_1 + b = 1$ and the Support Vector $(2, 1)$ lies on $w_1 x_1 + b = -1$. (you can use the other two svs as well).
Plugging these points, we get two equations with two unknowns.

$$w_1 + b = 1$$

$$2w1 + b = -1$$

Solving them gives us $w_1 = -2$, $b = 3$. So the final solution is $w_1 = -2$, $w_2 = 0$ 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 $\xi_i^* < 0$.)
Assume, for contradiction, that $\xi_i^* < 0$ for some i. Then we have

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

This means that setting $\xi_i^* = 0$ will still satisfy the constraints yet make the objective smaller, contradicting to the fact that ξ_i^* is the optimal solution. This proves that we must have $\xi_i^* \geq 0$ for all i when the penalty is the square of ξ_i.

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.
The lagrangian of this new problem is

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

where α_i's are the lagrange multipliers.

c. (8pts) Derive the dual of this problem following the same procedure as illustrated in the lecture. How does the dual problem differ from that of the standard SVM with hinge loss? In particular, which of the two formulations is more sensitive to outliers? Why?

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

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^N \alpha_i y_i = 0 \Rightarrow \sum_{i=1}^N \alpha_i y_i = 0$$

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

Plugging $\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$ *and* $\xi_i = \frac{\alpha_i}{2c}$ *into* L*, we have*

$$L(\alpha) = \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + c\sum_i \frac{\alpha_i^2}{4c^2} + \sum_i \alpha_i - \sum_i \frac{\alpha_i^2}{2c} - \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j + b\sum_i \alpha_i y_i$$

Rearranging the terms and dropping the last one (=0), we have:

$$L(\alpha) = -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \frac{1}{4c}\sum_i \alpha_i^2 + \sum_i \alpha_i$$

$$\text{subject to } \sum_i y_i \alpha_i = 0, \text{ and } \alpha_i \geq 0 \text{ for } i = 1, ..., N$$

There are several differences with the dual problem of the standard SVM. It has an additional α_i^2 *term. More importantly, it does not have the box constraint on* α_i*'s. (We didn't do this in class but it would be a good practice to go through the derivation for the original soft-margin svm to see where the box constraint comes from.) As a result, it is more sensitive to outliers than the standard SVM. This is intuitive because squared slack increases much faster for large slack values, so the impact of outliers tends to be stronger.*