1 Bias Variance Trade-off (10 Points)

Part (a) *[2 points]* The closed form solution for $\widehat{\boldsymbol{\beta}}_\lambda$ can be written as follows:

$$\widehat{\boldsymbol{\beta}}_\lambda = (X^\top X + \lambda I)^{-1} X^\top \mathbf{y} = (X^\top X + \lambda I)^{-1} X^\top (X\boldsymbol{\beta}^\star + \boldsymbol{\varepsilon})$$

Given the theorem about affine transformation of Gaussian random vectors, we can see that $\widehat{\boldsymbol{\beta}}_\lambda$ will be a Gaussian random vector with the following mean and variance:

$$\widehat{\boldsymbol{\beta}}_\lambda \sim \mathcal{N}\left((X^\top X + \lambda I)^{-1} X^\top X \boldsymbol{\beta}^\star, (X^\top X + \lambda I)^{-1} X^\top X (X^\top X + \lambda I)^{-1}\right).$$

Part (b) *[3 points]* Using part (a), we can write the bias as follows:

$$\mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda - \mathbf{x}^\top \boldsymbol{\beta}^\star] = \mathbf{x}^\top \mathbb{E}\left[\widehat{\boldsymbol{\beta}}_\lambda - \boldsymbol{\beta}^\star\right] = \mathbf{x}^\top \left((X^\top X + \lambda I)^{-1} X^\top X - I\right)\boldsymbol{\beta}^\star.$$

Part (c) *[3 points]* For the variance part, using the theorem about affine transformation of Gaussian random vectors again, we realize that $\mathbf{x}^\top(\widehat{\boldsymbol{\beta}}_\lambda - \mathbb{E}[\widehat{\boldsymbol{\beta}}_\lambda])$ is a zero-mean Gaussian random variable with variance $\mathbf{x}^\top (X^\top X + \lambda I)^{-1} X^\top X (X^\top X + \lambda I)^{-1} \mathbf{x} = \|X(X^\top X + \lambda I)^{-1}\mathbf{x}\|_2^2$. Thus, because the square of a Gaussian variable is a χ^2 random variable, we can use the mean of χ^2 random variable to conclude:

$$\mathbb{E}\left[\left(\mathbf{x}^\top(\widehat{\boldsymbol{\beta}}_\lambda - \mathbb{E}[\widehat{\boldsymbol{\beta}}_\lambda])\right)^2\right] = \|X(X^\top X + \lambda I)^{-1}\mathbf{x}\|_2^2.$$

Part (d) *[2 points]* The bias and variance trade-off can be written as:

$$\mathbb{E}\left[\left(\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda - \mathbf{x}^\top \boldsymbol{\beta}^\star\right)^2\right] = (\mathbf{x}^\top \left((X^\top X + \lambda I)^{-1} X^\top X - I\right)\boldsymbol{\beta}^\star)^2 + \|X(XX^\top + \lambda I)^{-1}\mathbf{x}\|_2^2 + \text{const}.$$

It is clear that as λ increases, the bias term increases and the variance term decreases.

2 Kernel Construction (15 Points)

Part (a) *[5 points]* Since K_1 and K_2 are symmetric kernel matrices, so is $K_3 = a_1 K_1 + a_2 K_2$. K_3 is also positive semidefinite: $v^T K_3 v = a_1 v^T K_1 v + a_2 v^T K_2 v \geq 0$ (because $a_1, a_2 \geq 0$ and K_1, K_2 are positive semi-definite). So k_3 is a valid kernel function.
Note: Extends to multiple kernels i.e. $k = \sum_{i=1}^p a_i k_i$ is a valid kernel if all $a_i \geq 0$ and all k_is are valid kernels.

Part (b) *[5 points]* $k_4(x, x') = f(x)f(x')$ where $f(\cdot)$ is a real valued function.
Symmetry: $K_4^T = [k_4(x_i, x_j)]^T = [k_4(x_j, x_i)] = f(x_j)f(x_i) = f(x_i)f(x_j) = [k_4(x_i, x_j)] = K_4$
Positive semi-definite: $v^T K_4 v = \sum_i \sum_j v_i f(x_i) f(x_j) v_j = \sum_i v_i f(x_i) \sum_j v_j f(x_j) = (\sum_i v_i f(x_i))^2 \geq 0$

Part (c) *[5 points]* $k_5(x, x') = k_1(x, x')k_2(x, x')$
For this part, we shall go with the basic definition of a valid kernel instead of Mercer's Theorem. So given the valid kernels $k_1(x, x') = \phi_1(x)^T \phi_1(x')$ and $k_2(x, x') = \phi_2(x)^T \phi_2(x')$, if we can find a feature vector transform ϕ_5 for k_5, such that $k_5(x, x') = \phi_5(x)^T \phi_5(x')$ then that qualifies k_5 as a valid kernel.
We first write the expression for $k_5(x, x')$:

$$k_5(x, x') = k_1(x, x')k_2(x, x') = \phi_1(x)^T \phi_1(x') \phi_2(x)^T \phi_2(x')$$

Now substituting ϕ_1 and ϕ_2 with their component-wise representation:

$$\begin{aligned}
k_5(x,x') &= \sum_{i=1}^{T} \phi_{1i}(x)\phi_{1i}(x') \sum_{j=1}^{T} \phi_{2j}(x)\phi_{2j}(x') \\
&= \sum_{i=1}^{T}\sum_{j=1}^{T} \phi_{1i}(x)\phi_{1i}(x')\phi_{2j}(x)\phi_{2j}(x') \\
&= \sum_{i=1}^{T}\sum_{j=1}^{T} (\phi_{1i}(x)\phi_{2j}(x))(\phi_{1i}(x')\phi_{2j}(x')) \\
&= \sum_{k=1}^{T^2} \phi_{5k}(x)\phi_{5k}(x') \\
&= \phi_5(x)^T\phi_5(x')
\end{aligned}$$

where $\phi_5(x)$ is a T^2 length feature transformation given by stacking all the possible component-wise products of the two feature transformations $\phi_1(x)$ and $\phi_2(x)$ i.e. $\phi_{5k}(x) = \phi_{1i}(x)\phi_{2j}(x)$ for each $k = (i-1)T + j$. Since $\phi_5(x)$ exists, $k_5(x,x')$ is a valid kernel.

3 Kernel Regression (15 Points)

Part (a) *[3 points]* The problem can be written as:

$$\begin{aligned}
\min_w \mathcal{L}(w) &= \min_w (y - Xw)^T(y - Xw) + \lambda w^T w \\
&= \min_w w^T X^T X w - 2y^T X w + \lambda w^T w
\end{aligned}$$

To minimize this, we differentiate w.r.t. w and set the derivative to zero:

$$\begin{aligned}
\nabla_w \mathcal{L} &= 2X^T X w - 2X^T y + 2\lambda w = 0 \\
\implies w^* &= (X^T X + \lambda I_D)^{-1} X^T y
\end{aligned}$$

Part (b) *[5 points]* Using the identity: $(P^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = PB^T(BPB^T + R)^{-1}$ with $R = I_N, B = \Phi$ and $P = \frac{I_D}{\lambda}$ in the result of Part (a) (with X replaced by transformed features Φ), we get:

$$\begin{aligned}
w^* &= \frac{I_D}{\lambda}\Phi^T(\Phi\frac{I_D}{\lambda}\Phi^T + I_N)^{-1} y \\
&= \Phi^T(\Phi\Phi^T + \lambda I_N)^{-1} y
\end{aligned}$$

Part (c) *[2 points]* To classify with w^*, we compute:

$$\begin{aligned}
\hat{y} &= w^{*T}\phi(x) \\
&= (\Phi^T(\Phi\Phi^T + \lambda I_N)^{-1} y)^T \phi(x) \\
&= y^T(\Phi\Phi^T + \lambda I_N)^{-1}\Phi\phi(x) \qquad \text{(because } \Phi\Phi^T + \lambda I_N \text{ is symmetric)}
\end{aligned}$$

Denoting $\Phi\Phi^T$ as K and $\Phi\phi(x)$ as $\kappa(x)$,

$$\hat{y} = y^T(K + \lambda I_N)^{-1}\kappa(x)$$

Part (d) *[5 points]*

- Training Complexity of Linear Ridge Regression: Training for linear ridge regression involves computing $w^* = (X^T X + \lambda I_D)^{-1} X^T y$. Note that X has dimensions $N \times D$ and y is $N \times 1$. So the computation complexities of various operations are:
  - $X^T X$: $O(ND^2)$
  - Add $X^T X$ and λI_D: $O(D^2)$
  - $X^T y$: $O(ND)$
  - Inverting $(X^T X + \lambda I_D)$ and multiplying with $X^T y$: $O(D^3)$

  Total: $O(ND^2 + D^3)$.

- Prediction Complexity of Linear Ridge Regression: Prediction requires outputting ${w^*}^T x$ which can be done in $O(D)$ time.

- Training Complexity of Kernel Ridge Regression: For kernel ridge regression, we can pre-compute the term $M = y^T (K + \lambda I_N)^{-1}$ during the training phase since it does not depend on the incoming test example. Since K is $N \times N$ and y is $N \times 1$, the complexity of the various operations are as follows:
  - Computing Kernel Matrix (K): Requires computing $\Phi\Phi^T$, but note that we don't need to compute $\phi(x)_{T\times 1}$ for this and hence the complexity doesn't depend on T. Let's denote the complexity of computing $k(x, z)$ as $O(k)$, then the total time to compute N^2 entries of matrix K can be done in $O(kN^2)$. For example, the linear kernel $K = XX^T$ takes $O(DN^2)$ to compute, the polynomial kernel $K = (1 + XX^T)^p$ takes about $O((D + \log p)N^2)$.
  - Invert $K + \lambda I_N$: $O(N^3)$.
  - Multiply y^T and $(K + \lambda I_N)^{-1}$: $O(N^2)$.

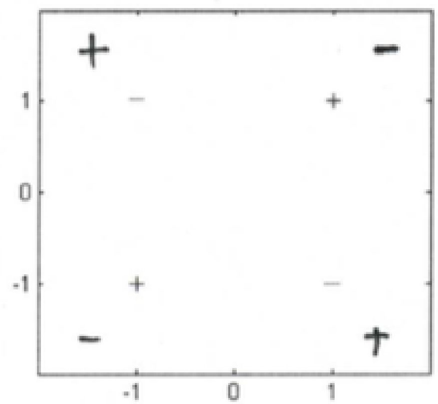
  Total: $O(kN^2 + N^3 + N^2) = O((k + N)N^2)$.

- Prediction Complexity of Kernel Ridge Regression: With M pre-computed, we need to compute $\hat{y} = M\kappa(x)$ for prediction. Since $\kappa(x)$ requires $O(kN)$ time, the total time required for classification is $O(kN + N) = O(kN)$.

4 Support Vector Machine (5 Points)

Consider a supervised learning problem in which the training examples are points in 2-dimensional space. The positive examples are $(1, 1)$ and $(-1, -1)$. The negative examples are $(1, -1)$ and $(-1, 1)$.

1. *[1 points]* No
2. *[1 points]* $w = (0, 0, 0, 1)^T$
3. *[1 points]* The solution is shown in the following figure:
4. *[2 points]* $1 + X_1 X_1' + X_2 X_2' + X_1 X_1' X_2 X_2'$



5 SVMs and the slack penalty C (15 Points)

1. *[3 points]* For large values of C, the penalty for misclassifying points is very high, so the decision boundary will perfectly separate the data if possible. See below for the boundary learned using libSVM and $C = 100000$.

COMMON MISTAKE 1: Some students drew straight lines, which would not be the result with a quadratic kernel.

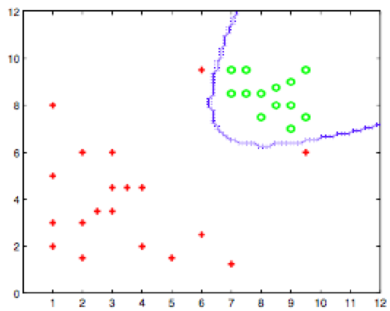
COMMON MISTAKE 2: Some students confused the effect of C and thought that a large C meant that the algorithm would be more tolerant of misclassifications.

2. *[3 points]* The classifier can maximize the margin between most of the points, while misclassifying a few points, because the penalty is so low. See below for the boundary learned by libSVM with $C = 0.00005$.

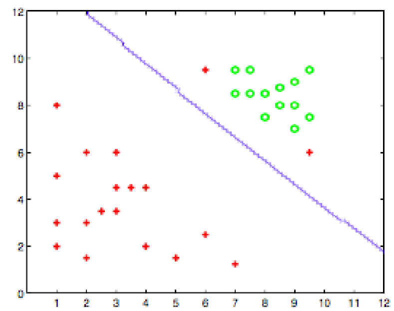
3. *[3 points]* We were warned not to trust any specific data point too much, so we prefer the solution where $C \approx 0$, because it maximizes the margin between the dominant clouds of points.

4. *[3 points]* We add the point circled below, which is correctly classified by the original classifier, and will not be a support vector.

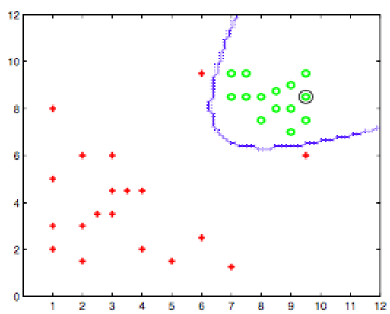
5. *[3 points]* Since C is very large, adding a point that would be incorrectly classified by the original boundary will force the boundary to move.



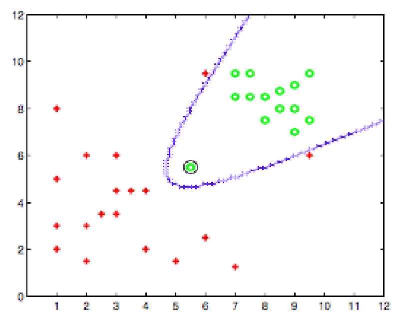
(a) Part 1



(b) Part 2



(c) Part 4



(d) Part 5

Figure 1: Draw your solutions here.