Solutions by **Andrew Lys** `andrewlys(at)u.e.`

1. 0/1 Loss vs Hinge Loss vs Square Loss

(a) (i) Let

$$S = \{((1,40),-1),((2,1),1),\ldots,((7,1),1),((2,-1),-1),\ldots,((6,-1),-1)\}$$

(ii) Letting $w = (0,1)$, we have that the decision boundary occurs at $y = \pm 1$, so all points are correctly classified except for the first point, i.e.

$$L_S^{01}(h_w) = \frac{1}{12} < 0.1$$

It is visually obvious that this is the only linear predictor in which only one point is misclassified. The only loss comes from $((1,40),-1)$, so the hinge loss is just

$$L_S^{\text{hinge}}(h_w) = \frac{1}{11}[1-(-1)(40)]_+ = \frac{41}{11}$$

(iii) We calculate the hinge loss minimizer via the methods of LaGrange Optimization.
Note that the only way we change w is by increasing the angle with the x-axis, since we are trying to minimize the loss added by the first point. Thus, in our hinge loss calculation, all the points below the x-axis will not add any more loss, so we may just disregard them, and restrict $w_1 \le 0$ and $w_2 \ge 0$
Our LaGrangian is as follows:

$$\mathcal{L} = \frac{1}{11}\langle (1,40),(w_1,w_2)\rangle + \frac{1}{11}\sum_{k=2}^{7} 1 - \langle (k,1),(w_1,w_2)\rangle - \lambda(w_1^2+w_2^2-1) - \mu_1 x - \mu_2 y$$

Solving this problem, we get:

$$w = (-1,0)$$

so our predictor is

$$\hat{h}^{\text{hinge}}(x) = \langle (-1,0), x\rangle$$

We can clearly see that this predictor correctly classifies $(1,40)$, but incorrectly classifies the rest of the positive points, but correctly classifies all of the negative points. Thus the 01 loss is:

$$L_S^{01}(\hat{h}^{\text{hinge}}) = \frac{6}{12} = 0.5$$

The hinge loss is given from the positive signs as:

$$\frac{1}{12}\sum_{k=2}^{7} 1-(-1)(k) = 2.75$$

2. Kernel Perceptron

(a) Base case:

$$w_0 = 0 \in \operatorname{lin}\{\phi(x_1),\ldots,\phi(x_m)\}$$

Inductive Step:
Suppose that $w_t \in \operatorname{lin}\{\phi(x_1),\ldots,\phi(x_m)\}$. If there is no mistake in the process, then $w_{t+1} = w_t$. If there is a mistake then

$$w_{t+1} = w_t - y_i\phi(x_i) \in \operatorname{lin}\{\phi(x_1),\ldots,\phi(x_m)\}$$

Thus, w_t is in the span of $\{\phi(x_1),\ldots,\phi(x_m)\}$.

(b) The updated algorithm is as follows:

```
$\alpha_0 \leftarrow 0$
$t \leftarrow 0$
**while** $\exists i : \operatorname{sign}\left(\sum_{j=1}^{m} \alpha_t[j]K(x_j,x_i)\right) \neq y_i$ **do**
    $\alpha_{t+1} = \alpha_t + y_i e_i$
    $t \leftarrow t+1$
**end while**
```

```
    return $\alpha_t$
```

(c) We focus on the while part of the iteration. We need to compute

$$G[i]\alpha_t \neq y_i$$

for $i = 1$ to m until there is a mistake.

$$G[i] = \begin{bmatrix} K(x_i, x_1) & K(x_i, x_2) & \dots & K(x_i, x_m) \end{bmatrix}$$

So for each i we have to make $\text{TIME}_k \cdot m$ calculations.

Assuming that arithmetic operations are $O(1)$, $G[i]\alpha_t$ takes m multiplication operations, and $m-1$ summation operations, and comparing to y_i is one operation, so our total runtime per each while check is:

$$O(\text{TIME}_k \cdot m + 2m) = O(\text{TIME}_k \cdot m)$$

In the worst case, each iteration has to search the entire data set for a mistake, so the runtime for the entire while loop is $O(\text{TIME}_k \cdot m^2)$.

Once we find a mistake, there are 2 arithmetic operations, so the overall runtime is $O(\text{TIME}_k \cdot m^2)$.

The challenge is done by pre-computing the matrix, which takes $O(\text{TIME}_k + m^2)$ storing it, requiring $O(m^2+m) = O(m^2)$, and working the while loop off of this stored matrix. This decreases the time complexity, per loop, to $O(m^2)$ and thus the overall time-complexity is $O(\text{TIME}_k + m^2)$.

(d) As we derived in homework 4, the number of mistakes, M_t, is bounded by $1/\gamma(S)^2$. $\gamma(S)$ was defined as:

$$\gamma(S) := \sup_w \min_{(x_i,y_i)\in S} \frac{y_i\langle w, \phi(x_i)\rangle}{\|w\|}$$

and assuming that all the feature vectors are norm 1. However, our feature vectors are not necessarily norm 1, so we normalize by dividing by $\|\phi(x_i)\|$ or $\sqrt{K(x_i,x_i)}$. So we actually have

$$\gamma(S) := \sup_w \min_{(x_i,y_i)\in S} \frac{y_i\langle w, \phi(x_i)\rangle}{\|w\|\sqrt{K(x_i,x_i)}}$$

By assumption, we have:

$$\gamma(S) \geq \min_{(x_i,y_i)\in S} \frac{y_i\langle w^*, \phi(x_i)\rangle}{\|w^*\|\sqrt{K(x_i,x_i)}} \geq \frac{\gamma}{\|w^*\| \max_i \sqrt{K(x_i,x_i)}}$$

Since the number of iterations, T is bounded by the number of mistakes, $\sup_t M_t$, since an iteration only occurs if a mistake occurs, we have:

$$T \leq \sup_t M_t \leq \frac{1}{\gamma(S)^2} \leq \frac{\|w^*\|^2 \max_i K(x_i,x_i)}{\gamma^2}$$

The time to compute a single iteration was given above as bounded by $O(\text{TIME}_k \cdot m^2)$ operations, and we only have to store $G[i]$ and α_t, which is a total of $O(m)$ memory. Thus, if the number of iterations is bounded by T_{max}, we have that the total runtime is bounded by:

$$O(\text{TIME}_k \cdot m^2 \cdot T_{max})$$

and the total memory space is bounded by

$$O(m)$$

(e) The perceptron algorithm returns w_T, which may be written as

$$\Phi^\intercal \alpha_T$$

Where α_T is the vector returned in part b. Thus, to compute $\text{sign}(\langle w_T, \phi(x)\rangle)$, we have to do the following:

$$\begin{aligned}\langle \Phi^\intercal \alpha_T, \phi(x)\rangle &= \alpha_T^\intercal \Phi \phi(x) \\ &= \alpha_T^\intercal \begin{bmatrix} \phi(x_1)^\intercal \\ \vdots \\ \phi(x_m)^\intercal \end{bmatrix} \phi(x) = \alpha_T^\intercal \begin{bmatrix} K(x_1,x) \\ \vdots \\ K(x_m,x) \end{bmatrix}\end{aligned}$$

Thus, we need to store α_t in our predictor instance, and when we predict, we need to compute $K(x_1,x)\dots,K(x_m,x)$, and dot it with α_T. Computing this vector takes $O(m \cdot \text{TIME}_k)$, and requires us to store $K(x_1,\cdot),\dots K(x_m,\cdot)$. This means that we store m floats, and m pointers, for a total of $O(m)$ memory space. To compute the prediction runtime, we do m kernel operations, for a total of $O(m \cdot \text{TIME}_k)$, but then we do $2m$ arithmetic operations, for a total of

$$O(m \cdot \text{TIME}_k)$$

run time complexity.

3. Kernel Ridge Regression

(a) Note that we can write:

$$w(\alpha) = \sum_{i=1}^{m} \alpha_i \phi(x_i) = \Phi^\intercal \alpha$$

$$\begin{aligned} L_{S,\lambda}(\alpha) &= \frac{1}{m}\|\Phi w(\alpha) - y\|^2 + \lambda \|w(\alpha)\|^2/2 \\ &= \frac{1}{m}\|\Phi\Phi^\intercal \alpha - y\|^2 + \lambda\|\Phi^\intercal \alpha\|^2/2 \\ &= \frac{1}{m}\|G\alpha - y\|^2 + \lambda \frac{\alpha^\intercal \Phi \Phi^\intercal \alpha}{2} \\ &= \frac{1}{m}\|G\alpha - y\|^2 + \frac{\lambda}{2}\alpha^\intercal G \alpha \end{aligned}$$

(b) With elementary matrix calculus, we can calculate the derivative of the above expression with respect to α.
Recall:

$$\frac{d}{du}\|u\|^2 = 2u$$

$$\frac{d}{d\alpha}\alpha^\intercal G\alpha = 2G\alpha$$

Thus, we have:

$$\begin{aligned} \frac{d}{d\alpha}L_{S,\lambda} &= \frac{d}{d\alpha}\left(\frac{1}{m}\|G\alpha - y\|^2 + \frac{\lambda}{2}\alpha^\intercal G\alpha\right) \\ &= \frac{1}{m}\frac{d}{d\alpha}\|G\alpha - y\|^2 + \frac{\lambda}{2}\frac{d}{d\alpha}\alpha^\intercal G\alpha \\ &= \frac{1}{m}2G^\intercal(G\alpha - y) + \lambda G\alpha \\ &= \frac{2}{m}G^\intercal G\alpha - \frac{2}{m}G^\intercal y + \lambda G\alpha \\ &= \frac{2}{m}G^\intercal G\alpha + \lambda G\alpha - \frac{2}{m}G^\intercal y = 0 \\ \frac{2}{m}G^\intercal y &= \frac{2}{m}G^\intercal G\alpha + \lambda G\alpha \\ \frac{2}{m}Gy &= \left(\frac{2}{m}G^2 + \lambda G\right)\alpha \\ \implies \alpha &= \left(\frac{2}{m}G^2 + \lambda G\right)^{-1}\frac{2}{m}Gy \end{aligned}$$

Since G is positive definite, we have that G^{-1} exists, so we may simplify to:

$$\alpha = \left(\frac{2}{m}G + \lambda I_m\right)^{-1}\frac{2}{m}y$$

Thus, this is the optimal α, and we have:

$$\hat{\alpha} = \left(G + \frac{m}{2}\lambda I_m\right)^{-1} y$$

(c) Using the same idea as in (e), we have:

$$\begin{aligned}
\langle \hat{w}_\lambda, \phi(x) \rangle &= \langle \Phi\hat{\alpha}, \phi(x) \rangle \\
&= \hat{\alpha}^\intercal \Phi\phi(x) \\
&= \hat{\alpha}^\intercal \begin{bmatrix} \phi(x_1)^\intercal \\ \vdots \\ \phi(x_m)^\intercal \end{bmatrix} \phi(x) = \hat{\alpha}^\intercal \begin{bmatrix} K(x_1, x) \\ \vdots \\ K(x_m, x) \end{bmatrix} \\
&= \begin{bmatrix} K(x_1, x) & \dots & K(x_m, x) \end{bmatrix} \left(\frac{2}{m} G^2 + \lambda G \right)^{-1} \frac{2}{m} G y
\end{aligned}$$