Problem 1

Note: I have used my own definition of ϕ as a vector in $\mathbb{R}^n$ for parts $(a)-(f)$.

(a) $K(x,z)$ is a kernel. We can see this by writing $K_1(x,z) = \phi_1^T(x)\phi_1(z)$ and $K_2(x,z) = \phi_2^T(x)\phi_2(z)$. Then

$$\begin{aligned} K(x,z) &= K_1(x,z) + K_2(x,z) \\ &= \phi_1^T(x)\phi_1(z) + \phi_1^T(x)\phi_1(z) \\ &= \begin{pmatrix}\phi_1^T(x) & \phi_2^T(x)\end{pmatrix}\begin{pmatrix}\phi_1(z) \\ \phi_2(z)\end{pmatrix} \\ &= \tilde{\phi}^T(x)\tilde{\phi}(z) \end{aligned}$$

where we have defined the new vector $\tilde{\phi}(x) = (\phi_1(x)\ \phi_2(x))^T$, i.e. $\tilde{\phi}(x) \in \mathbb{R}^{2n}$.

(b) $K(x,z) = K_1(x,z) - K_2(x,z)$ is not a Kernel function. This can be shown by a counter-example: take $K_1(x,z) = 0$, then $K(x,z) = -K_2(x,z)$, which is the negative of a Kernel function, and which by definition cannot correspond to a positive semidefinite matrix.

(c) $K(x,z) = aK_1(x,z)$ is a Kernel. We can see this by writing $K_1(x,z) = \phi_1^T(x)\phi_1(z)$, and defining the related vector $\tilde{\phi}(x) = \sqrt{a}\phi_1(x)$, which is a real vector because $a > 0$. It should then be clear that $K(x,z) = \tilde{\phi}^T(x)\tilde{\phi}(z)$.

(d) $K(x,z) = -aK_1(x,z)$ is not a Kernel. This is because we can write it as the negative of a true Kernel (following part (c) above, $K(x,z) = -\tilde{\phi}^T(x)\tilde{\phi}(z)$, where $\tilde{\phi}^T(x)\tilde{\phi}(z)$ *is* a valid Kernel), at which point the corresponding Kernel matrix is not positive semi-definite.

(e) $K(x,z) = K_1(x,z)K_2(x,z)$ is a valid Kernel. To see why, let me write each Kernel explicitly (repeated indices are being summed over , as explicitly shown in the for K_1):

$$\begin{aligned} K_1(x,z) &\equiv \phi^T(x)\phi(z) \equiv \sum_\mu \phi_\mu(x)\phi_\mu(z) \equiv \phi_\mu(x)\phi_\mu(z) \\ K_2(x,z) &\equiv \psi^T(x)\psi(z) \equiv \psi_\nu(x)\psi_\nu(z) \end{aligned}$$

Then, we see that

$$\begin{aligned} K(x,z) &= \phi_\mu(x)\phi_\mu(z)\psi_\nu(x)\psi_\nu(z) \\ &= \psi_\nu(x)\phi_\mu(x)\phi_\mu(z)\psi_\nu(z) \\ &= \Lambda_{\nu\mu}(x)\Lambda_{\mu\nu}(z) \\ &\equiv \mathrm{Tr}[\Lambda^T(x)\Lambda(z)] \\ &= \eta^T(x)\eta(z) \end{aligned}$$

where in the last line we have noted that the Trace of the product of two matrices can be written as the inner product (a.k.a. dot product) of two vectors. To be explicit, consider for simplicity a 2×2 matrix:

$$\Lambda(x) = \begin{pmatrix}\Lambda_{11}(x) & \Lambda_{12}(x) \\ \Lambda_{21}(x) & \Lambda_{22}(x)\end{pmatrix} \tag{1}$$

Then,

$$\begin{aligned}
\mathrm{Tr}[\Lambda^T(x)\Lambda(z)] &= \mathrm{Tr}\left[\begin{pmatrix}\Lambda_{11}(x) & \Lambda_{21}(x)\\ \Lambda_{12}(x) & \Lambda_{22}(x)\end{pmatrix}\begin{pmatrix}\Lambda_{11}(z) & \Lambda_{21}(z)\\ \Lambda_{12}(z) & \Lambda_{22}(z)\end{pmatrix}\right]\\
&= \mathrm{Tr}\left[\begin{pmatrix}\Lambda_{11}(x)\Lambda_{11}(z)+\Lambda_{21}(x)\Lambda_{12}(z) & \dots\\ \dots & \Lambda_{22}(x)\Lambda_{22}(z)+\Lambda_{12}(x)\Lambda_{21}(z)\end{pmatrix}\right]\\
&= \Lambda_{11}(x)\Lambda_{11}(z)+\Lambda_{21}(x)\Lambda_{12}(z)+\Lambda_{22}(x)\Lambda_{22}(z)+\Lambda_{12}(x)\Lambda_{21}(z)
\end{aligned}$$

This final expression can be written as an inner product of two vectors: $\eta^T(x)\eta(z)$, where

$$\eta^T(x) = (\Lambda_{11}(x), \Lambda_{12}(x), \Lambda_{21}(x), \Lambda_{22}(x))$$

.

(f) $K(x,z) = f(x)f(z)$ is a Kernel. This is almost trivial to prove: $f(x)$ can be considered as a one component vector, $\vec{f}(x)$, in which case it is clear that $K(x,z) = f^T(x)f(z)$. This ensures that K is a Mercer Kernel because its matrix representation is positive semidefinite.

(g) $K(x,z) = K_3(\phi(x),\phi(z))$ is a Kernel. Writing $K_3(x,z) = \xi^T(u)\xi(v)$, where $u,v \in \mathbb{R}^d$, we find that

$$K(x,z) = \xi^T(\phi(x))\xi(\phi(z)) = \chi^T(x)\chi(z)$$

where I have simply defined a new real function $\chi(x) = \xi(\phi(x))$ where $\chi \in \mathbb{R}^d$.

(h) $K(x,z)$ is a Kernel. Define

$$p(x) = \sum_n a_n x^n \tag{2}$$

then

$$p(K_1(x,z)) = \sum_n a_n K_1^n(x,z) \tag{3}$$

By successive use of part **(e)**, we see that every single term in this polynomial can be written as a single Kernel, i.e. $K_1^n(x,z) \equiv K_n(x,z)$. Then, using **(c)** we can absorb each positive coefficient into the definition of the polynomial, $a_nK_n(x,z) = K_m(x,z)$. Finally, using **(a)**, we see that the sum of several Kernels is itself a Kernel. $K(x,z)$ is thus a valid Kernel.

Problem 2

(a) Note that with the perceptron algorithm update rule, and given $\theta^{(0)} = \vec{0}$, we find that μ'th component of the vector $\theta^{(1)}$ is

$$\theta^{(1)}_\mu = \alpha(y^{(1)} - 1)\phi^{(1)}_\mu \equiv \alpha^{(1)}\phi_\mu(x^{(1)})$$

where I have defined an i-dependent coefficient $\alpha^{(i)}$ (which takes values $\alpha^{(i)} = 0, \pm 2$ if we use $g(z) = \text{sign}(z)$). It is clear that with this update rule, the i'th update to θ takes the implicit form

$$\theta^{(i)} = \sum_{j=1}^{i-1} \alpha^{(j)}\phi_\mu(x^{(j)})$$

With this representation, our sole task will be to determine the coefficients $\alpha^{(j)}$ from the training set.

(b) To make a new prediction with input $x^{(i+1)}$, we must compute

$$g\left(\theta^{(i)^T}\phi(x^{(i+1)})\right) = \text{sign}\left(\sum_{j=1}^{i} \alpha^{(j)}\phi^T(x^{(j)})\phi(x^{(i+1)})\right) = \text{sign}\left(\sum_{j=1}^{i} \alpha^{(j)}K(x^{(j)}, x^{(i+1)})\right) \tag{4}$$

where I have defined the Kernel function $K(x, z)$ where $x, z \in \mathbb{R}^n$ are finite dimensional vectors corresponding to the raw input features. Note that having already computed the coefficients $\alpha^{(j)}$, we must only compute the Kernel functions, which can (apparently!?) be done efficiently.

(c) The update rule we must now implement involves computing the coefficients $\alpha^{(j)}$. The update rule now becomes:

For each new training example, compute :

$$\alpha^{(i+1)} = \alpha\left[y^{(i+1)} - \text{sign}\left(\sum_{j=1}^{i} \alpha^{(j)}K(x^{(j)}, x^{(i+1)})\right)\right]$$

where the sum over j is a sum over the previous i training examples. The result here is simply a number to be stored $(0, \pm 2)$.

Problem 3

All Matlab Code is written in the attached pages.

(a) The error rate for the full training set is 0.0163.

(b) The 5 tokens most indicative of the SPAM class are in descending order,

1. 616: httpaddr
2. 1210: spam
3. 1357: unsubscrib
4. 394: ebai
5. 1369: valet

(c) The errors for different training set sizes are shown in the table below. The largest training sets give the smallest error, although it is interesting that going from 1400 to 2144 emails in the training sample doesn't provide a performance boost.

Training Set Size	Error rate
50	0.0387
100	0.0262
200	0.0262
400	0.0187
800	0.0175
1400	0.0163
2144	0.0163

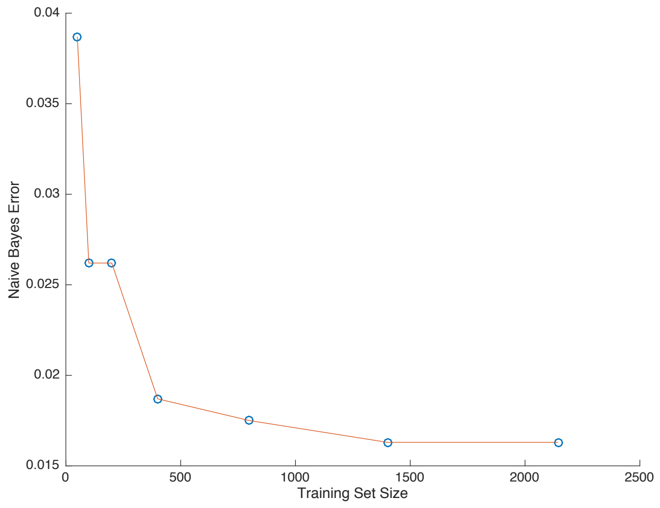

Figure 1: Learning curve for our implementation of the Naive Bayes algorithm on different training set sizes.

(d) The errors for different training set sizes upon implementing the SVM are shown here. The largest training set gives the smallest error.

Problem 3 continued on next page. . .

Training Set Size	Error rate
50	0.0525
100	0.0312
200	0.0125
400	0.0150
800	0.0125
1400	0.0100
2144	0.0063

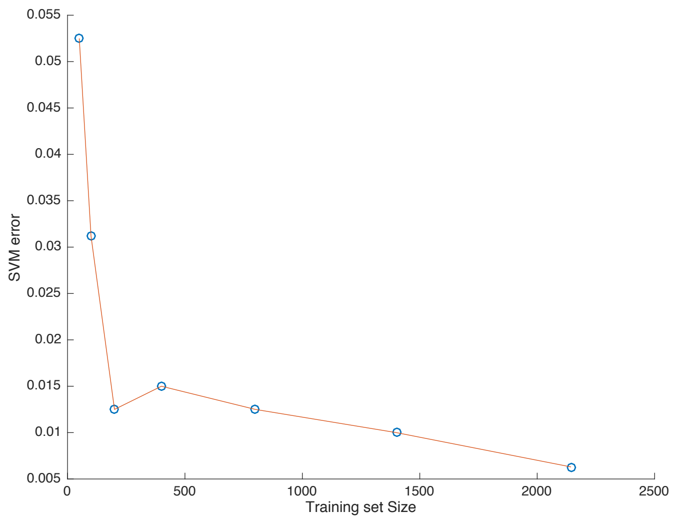

Figure 2: The learning curve for our implementation of the SVM.

(e) While Naive Bayes initially outperforms the SVM, it significantly underperforms for the largest data sets. This is most obvious if one plots the errors vs. training sample size on a log-log plot as shown in Fig. **??** . There is some evidence of a power law scaling between the error rate and training sample size for both algorithms, with the larger slope of the SVM (blue data) indicating its better performance for large samples.

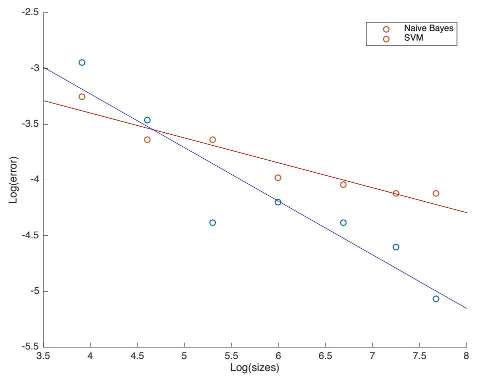

Figure 3: Log-Log plot comparing the error rates ϵ for Naive Bayes algorithm(red) and SVM (blue) vs. system size N. There is some evidence for power law scaling (i.e. error rate, $\epsilon \sim a(N)^{-p}$, with a larger fitted p for the SVM ($p = 0.481$) vs. Naive Bayes ($p = 0.224$).

Problem 4

(a)The statement is true. Assume that H_1 shatters a set S of dimension n. Then $\mathrm{VC}(H_1) = n$. Now because $H_1 \subseteq H_2$, it is clear that H_2 also shatters S. Hence, it is necessary that $\mathrm{VC}(H_2) \geq n$, i.e. $\mathrm{VC}(H_2) \geq \mathrm{VC}(H_1)$.

(b) The statement is true. As is hinted by the problem, we can prove the statement by induction. Thus, consider the set $H_1 = H_2 \cup \{h_1\}$. Now let us assume that H_1 shatters a set S_1 of dimension n. Consider a specific labeling on the set S_1, which is correctly classified by the hypothesis h_1. It is always possible to find a second hypothesis h_2 which classifies all points within the same labeling but one point (call that point x). The set of points $\tilde{S} = S_1 \setminus \{x\}$ is then shattered by a set of hypotheses $\tilde{H}$ and so

$$\mathrm{VC}(\tilde{H}) \geq n - 1 \tag{5}$$

Because h_1 and h_2 both classify $\tilde{S}$ correctly, and $h_1 \notin H_2$, it follows that $\tilde{H} \subseteq H_2$, so from part (a) we have

$$\mathrm{VC}(\tilde{H}) \leq \mathrm{VC}(H_2)$$

Using Eq. 5, we have

$$\begin{aligned} &\mathrm{VC}(H_2) \geq n - 1 \\ \implies &\mathrm{VC}(H_2) + 1 \geq n \\ &\text{i.e. } \mathrm{VC}(H_1) \leq \mathrm{VC}(H_2) + 1 \end{aligned}$$

The rest of the proof follows by induction.

(c) For $H_1 = H_2 \cup H_3$, **it is false that** $\mathrm{VC}(H_1) \leq \mathrm{VC}(H_2) + \mathrm{VC}(H_3)$. A simple counter example can suffice. Let $H_1 = \{h_1\}$ and $H_2 = \{h_2\}$, with $h_1(x) = 1$ and $h_2(x) = -1$. By itself, a single line cannot classify any finite number of points, hence we have $\mathrm{VC}(H_1) = \mathrm{VC}(H_2) = 0$. However these two lines together can in fact classify a single point, so we have $\mathrm{VC}(H_3 = \{h_1, h_2\}) = 1$. Hence the assertion is disproved.

Problem 5

(a) Because the classification is binary, an error is only possible in the corrupted distribution when there is initial error which is not flipped, or no initial error followed by a corruption. Thus we have

$$\varepsilon_\tau(h) = \varepsilon_0(h)\,[1-\tau] + [1-\varepsilon_0(h)]\,\tau$$

Simple rearrangement gives

$$\varepsilon_0(h) = \frac{\varepsilon_\tau(h)-\tau}{1-2\tau} \tag{6}$$

(b) Using the formula above, along with the Hoeffding inequality + the union bound to obtain a uniform convergence result on our training set, i.e.

$$P(\neg h \in \mathcal{H}.|\varepsilon_\tau(h_i) - \hat{\varepsilon}_\tau(h_i)| > \gamma_\tau) \geq 1 - 2|H|\exp\left(-2\gamma_\tau^2 m\right),$$

we then find the following:

$$\begin{aligned}
\varepsilon_0(\hat{h}) &= \frac{\varepsilon_\tau(\hat{h})-\tau}{1-2\tau} \\
&\leq \frac{\hat{\varepsilon}_\tau(\hat{h})+\gamma_\tau-\tau}{1-2\tau} \\
&\leq \frac{\hat{\varepsilon}_\tau(h^*)+\gamma_\tau-\tau}{1-2\tau} \\
&\leq \frac{\varepsilon_\tau(h^*)+2\gamma_\tau-\tau}{1-2\tau} \\
&= \varepsilon_0(h^*) + \frac{2\gamma_\tau}{1-2\tau}
\end{aligned}$$

where the first four steps above are as outlined in the lecture notes (notes 4, page 7). This occurs with probability $1-\delta$. So we have that for

$$\varepsilon_0(\hat{h}) \leq \varepsilon_0(h^*) + 2\gamma \tag{7}$$

where $\gamma = \gamma_\tau/(1-2\tau)$, we require

$$\begin{aligned}
m &\geq \frac{1}{2\gamma_\tau^2}\log\frac{2|H|}{\delta} \\
\text{i.e.}\quad m &\geq \frac{1}{2\gamma^2(1-2\tau)^2}\log\frac{2|H|}{\delta}
\end{aligned} \tag{8}$$

(c) As τ approaches 0.5, the training set is entirely random, and of cannot provide any predictive value. It is therefore sensible that a divergent number of training examples are necessary - training on a completely random data set is a useless proposition!