1. **[16 points] Least Squares**

   As described in class, in least squares regression we have a cost function:

   $$J(\theta) = \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 = (X\theta - \vec{y})^T (X\theta - \vec{y})$$

   The goal of least squares regression is to find θ such that we minimize $J(\theta)$ given the training data.

   Let's say that we had an original set of n features, so that the training inputs were represented by the design matrix $X \in \mathbb{R}^{m \times (n+1)}$. However, we now gain access to one additional feature for every example. As a result, we now have an additional vector of features $\vec{v} \in \mathbb{R}^{m \times 1}$ for our training set that we wish to include in our regression. We can do this by creating a new design matrix: $\widetilde{X} = [X\ \vec{v}] \in \mathbb{R}^{m \times (n+2)}$.

   Therefore the new parameter vector is $\theta_{new} = \begin{pmatrix} \theta \\ p \end{pmatrix}$ where $p \in \mathbb{R}$ is the parameter corresponding to the new feature vector $\vec{v}$.

   Note: For mathematical simplicity, throughout this problem you can assume that $X^T X = I \in \mathbb{R}^{(n+1)\times(n+1)}$ and $\widetilde{X}^T \widetilde{X} = I \in \mathbb{R}^{(n+2)\times(n+2)}, \vec{v}^T \vec{v} = 1$. This is called an *orthonormality* assumption – specifically, the columns of $\widetilde{X}$ are orthonormal. The conclusions of the problem hold even if we do not make this assumption, but this will make your derivations easier.

   (a) [2 points] Let $\hat{\theta} = \arg\min_\theta J(\theta)$ be the minimizer of the original least squares objective (using the original design matrix X). Using the orthornormality assumption, show that $J(\hat{\theta}) = (XX^T\vec{y} - \vec{y})^T(XX^T\vec{y} - \vec{y})$. I.e., show that this is the value of $\min_\theta J(\theta)$ (the value of the objective at the minimum).

   Answer: We know from lecture that the least squares minimizer is $\hat{\theta} = (X^T X)^{-1} X^T \vec{y}$ but because of the orthonormality assumption, this simplifies to $\hat{\theta} = X^T \vec{y}$. Substituting this expression into the normal equation for $J(\theta)$ gives the final expression $J(\hat{\theta}) = (XX^T\vec{y} - \vec{y})^T(XX^T\vec{y} - \vec{y})$.

(b) [5 points] Now let $\hat{\theta}_{new}$ be the minimizer for $\widetilde{J}(\theta_{new}) = (\widetilde{X}\theta_{new} - \vec{y})^T(\widetilde{X}\theta_{new} - \vec{y})$. Find the new minimized objective $\widetilde{J}(\hat{\theta}_{new})$ and write this expression in the form: $\widetilde{J}(\hat{\theta}_{new}) = J(\hat{\theta}) + f(X, \vec{v}, \vec{y})$ where $J(\hat{\theta})$ is as derived in part (a) and f is some function of $X, \vec{v}$, and $\vec{y}$.

Answer: Just like we had in part (a), the minimizer for the new objective is $\hat{\theta}_{new} = \widetilde{X}^T\vec{y}$. Now we solve for the new minimized objective:

$$\begin{aligned}
\widetilde{J}(\hat{\theta}_{new}) &= (\widetilde{X}\hat{\theta}_{new} - \vec{y})^T(\widetilde{X}\hat{\theta}_{new} - \vec{y}) \\
&= (\widetilde{X}\widetilde{X}^T\vec{y} - \vec{y})^T(\widetilde{X}\widetilde{X}^T\vec{y} - \vec{y}) \\
&= ((XX^T + \vec{v}\vec{v}^T)\vec{y} - \vec{y})^T((XX^T + \vec{v}\vec{v}^T)\vec{y} - \vec{y}) \\
&= ((XX^T\vec{y} - \vec{y}) + \vec{v}\vec{v}^T\vec{y})^T((XX^T\vec{y} - \vec{y}) + \vec{v}\vec{v}^T\vec{y}) \\
&= (XX^T\vec{y} - \vec{y})^T(XX^T\vec{y} - \vec{y}) + 2(XX^T\vec{y} - \vec{y})^T(\vec{v}\vec{v}^T\vec{y}) + (\vec{v}\vec{v}^T\vec{y})^T(\vec{v}\vec{v}^T\vec{y}) \\
&= J(\hat{\theta}) + 2(XX^T\vec{y} - \vec{y})^T(\vec{v}\vec{v}^T\vec{y}) + (\vec{v}\vec{v}^T\vec{y})^T(\vec{v}\vec{v}^T\vec{y})
\end{aligned} \tag{1}$$

(c) [6 points] Prove that the optimal objective value does not increase upon adding a feature to the design matrix. That is, show $\widetilde{J}(\hat{\theta}_{new}) \leq J(\hat{\theta})$.

Answer: Using the final result of part (b), we can continue simplifying the expression for $J(\hat{\theta}_{new})$ as follows:

$$\begin{aligned}
\widetilde{J}(\hat{\theta}_{new}) &= J(\hat{\theta}) + 2(XX^T\vec{y} - \vec{y})^T(vv^T\vec{y}) + (vv^T\vec{y})^T(vv^T\vec{y}) \\
&= J(\hat{\theta}) + 2(XX^T\vec{y})^T(vv^T\vec{y}) - 2\vec{y}^T(vv^T\vec{y}) + (vv^T\vec{y})^T(vv^T\vec{y}) \\
&= J(\hat{\theta}) + 2(\vec{y}^TXX^Tvv^T\vec{y}) - 2(\vec{y}^Tvv^T\vec{y}) + (\vec{y}^Tvv^Tvv^T\vec{y}) \\
&= J(\hat{\theta}) - \vec{y}^Tvv^T\vec{y} \\
&= J(\hat{\theta}) - (v^T\vec{y})^2 \\
&\leq J(\hat{\theta})
\end{aligned} \tag{2}$$

From the third to last equality to the second to last equality, we use the two facts that $X^Tv = 0$ and $v^Tv = 1$.

The proof is complete.

(d) [3 points] Does the above result show that if we keep increasing the number of features, we can always get a model that generalizes better than a model with fewer features? Explain why or why not.

Answer: The result shows that we can either maintain or decrease the minimized square error objective by adding more features. However, remember that the error objective is computed only on the training samples and not the true data distribution. As a result, reducing training error does not guarantee a reduction in error on the true distribution. In fact, after a certain point adding features will likely lead to overfitting, increasing our generalization error. Therefore, adding features does not actually always result in a model that generalizes better.

2. **[14 points] Decision Boundaries for Generative Models**

(a) [7 points] Consider the *multinomial event model* of Naive Bayes. Our goal in this problem is to show that this is a linear classifier.

For a given text document x, let $c_1, ..., c_V$ indicate the number of times each word (out of V words) appears in the document. Thus, $c_i \in \{0, 1, 2, \ldots\}$ counts the occurrences of word i. Recall that the Naive Bayes model uses parameters $\phi_y = p(y = 1)$, $\phi_{i|y=1} = p(\text{word i appears in a specific document position} \mid y = 1)$ and $\phi_{i|y=0} = p(\text{word i appears in a specific document position} \mid y = 0)$.

We say a classifier is linear if it assigns a label $y = 1$ using a decision rule of the form

$$\sum_{i=1}^{V} w_i c_i + b \geq 0$$

I.e., the classifier predicts "$y = 1$" if $\sum_{i=1}^{V} w_i c_i + b \geq 0$, and predicts $y = 0$ otherwise.

Show that Naive Bayes is a linear classifier, and clearly state the values of w_i and b in terms of the Naive Bayes parameters. (Don't worry about whether the decision rule uses "$\geq$" or "$>$.") Hint: consider using log-probabilities.

Answer: The decision boundary for Naive Bayes can be stated as

$$\begin{aligned}
P(y = 1|c; \Phi) &> P(y = 0|c; \Phi) \\
\log p(y = 1|c; \Phi) &> \log p(y = 0|c; \Phi) \\
\log p(y = 1|c; \Phi) - \log p(y = 0|c; \Phi) &> 0 \\
\log \frac{p(y = 1|c; \Phi)}{p(y = 0|c; \Phi)} &> 0 \\
\log \frac{p(y = 1) \prod_{i=1}^{V} p(E_i|y = 1)^{c_i}}{p(y = 0) \prod_{i=1}^{V} p(E_i|y = 0)^{c_i}} &> 0 \\
\log \frac{p(y = 1)}{p(y = 0)} + \sum_{i=1}^{V} \log \frac{p(E_i|y = 1)^{c_i}}{p(E_i|y = 0)^{c_i}} &> 0 \\
\log \frac{\phi_y}{1 - \phi_y} + \sum_{i=1}^{V} c_i \log \frac{\phi_{i|y=1}}{\phi_{i|y=0}} &> 0
\end{aligned}$$

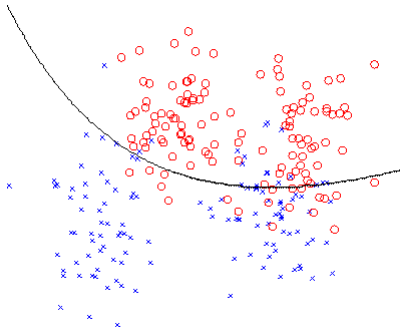
Thus, Naive Bayes is a linear classifier with

$$\begin{aligned}
w_i &= \log \frac{\phi_{i|y=1}}{\phi_{i|y=0}} \\
b &= \log \frac{\phi_y}{1 - \phi_y}
\end{aligned}$$

[extra space for 2 (a)]

(b) [7 points] In Problem Set 1, you showed that Gaussian Discriminant Analysis (GDA) is a linear classifier. In this problem, we will show that a modified version of GDA has a quadratic decision boundary.

Recall that GDA models $p(x|y)$ using a multivariate normal distribution, where $(x|y=0) \sim \mathcal{N}(\mu_0, \Sigma)$ and $(x|y=1) \sim \mathcal{N}(\mu_1, \Sigma)$, where we used the same Σ for both Gaussians. For this question, we will instead use two covariance matrices Σ_0, Σ_1 for the two labels. So, $(x|y=0) \sim \mathcal{N}(\mu_0, \Sigma_0)$ and $(x|y=1) \sim \mathcal{N}(\mu_1, \Sigma_1)$.



The model distributions can now be written as:

$$
\begin{aligned}
p(y) &= \phi^y (1-\phi)^{1-y} \\
p(x|y=0) &= \frac{1}{(2\pi)^{n/2}|\Sigma_0|^{1/2}} \exp\Big(-\frac{1}{2}(x-\mu_0)^T \Sigma_0^{-1}(x-\mu_0)\Big) \\
p(x|y=1) &= \frac{1}{(2\pi)^{n/2}|\Sigma_1|^{1/2}} \exp\Big(-\frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1)\Big)
\end{aligned}
$$

Let's follow a binary decision rule, where we predict $y=1$ if $p(y=1|x) \geq p(y=0|x)$, and $y=0$ otherwise. Show that if $\Sigma_0 \neq \Sigma_1$, then the separating boundary is quadratic in x.

That is, simplify the decision rule "$p(y=1|x) \geq p(y=0|x)$" to the form "$x^T A x + B^T x + C \geq 0$" (supposing that $x \in \mathbb{R}^{n+1}$), for some $A \in \mathbb{R}^{(n+1)\times(n+1)}$, $B \in \mathbb{R}^{n+1}$, $C \in \mathbb{R}$ and $A \neq 0$. Please clearly state your values for A, B and C.

[extra space for 2 (b)]

Answer: Examining the log-probabilities yields

$\log p(y=1|x) \geq \log p(y=0|x)$

$$
\begin{aligned}
0 \leq & \log\Big(\frac{p(y=1|x)}{p(y=0|x)}\Big) \\
0 \leq & \log\Big(\frac{p(y=1)p(x|y=1)}{p(y=0)p(x|y=0)}\Big) \\
0 \leq & \log\Big(\frac{\phi}{1-\phi}\Big) - \log\Big(\frac{|\Sigma_1|^{1/2}}{|\Sigma_0|^{1/2}}\Big) \\
& - \frac{1}{2}\Big((x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) - (x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)\Big) \\
0 \leq & -\frac{1}{2}\Big(x^T(\Sigma_1^{-1}-\Sigma_0^{-1})x - 2(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1})x \\
& + \mu_1^T\Sigma_1^{-1}\mu_1 - \mu_0^T\Sigma_0^{-1}\mu_0\Big) + \log\Big(\frac{\phi}{1-\phi}\Big) - \log\Big(\frac{|\Sigma_1|^{1/2}}{|\Sigma_0|^{1/2}}\Big) \\
0 \leq & x^T\Big(\frac{1}{2}(\Sigma_0^{-1}-\Sigma_1^{-1})\Big)x + \Big(\mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1}\Big)x \\
& + \log\Big(\frac{\phi}{1-\phi}\Big) + \log\Big(\frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}}\Big) + \frac{1}{2}\Big(\mu_0^T\Sigma_0^{-1}\mu_0 - \mu_1^T\Sigma_1^{-1}\mu_1\Big)
\end{aligned}
$$

From the above, we see that $A = \frac{1}{2}(\Sigma_0^{-1} - \Sigma_1^{-1})$, $B = \mu_1^T\Sigma_1^{-1} - \mu_0^T\Sigma_0^{-1}$, and $C = \log(\frac{\phi}{1-\phi}) + \log(\frac{|\Sigma_0|^{1/2}}{|\Sigma_1|^{1/2}}) + \frac{1}{2}(\mu_0^T\Sigma_0^{-1}\mu_0 - \mu_1^T\Sigma_1^{-1}\mu_1)$. Furthermore, $A \neq 0$ since $\Sigma_0 \neq \Sigma_1$ implies that $\Sigma_0^{-1} - \Sigma_1^{-1} \neq 0$. Therefore, the decision boundary is quadratic.

3. **[18 points] Generalized Linear Models**
In this problem you will build a Generalized Linear Model (GLM) for a response variable y (taking on values $\{1,2,3,\dots\}$), whose distribution (parameterized by ϕ) is modeled as:

$$p(y;\phi) = (1-\phi)^{y-1}\phi$$

This distribution is known as the *geometric distribution*, and is used to model network connections and many other problems.

(a) i. [5 points] Show that the geometric distribution is an exponential family distribution. You should explicitly specify $b(y)$, η, $T(y)$, $\alpha(\eta)$. Also specify what ϕ is in terms of η.
Answer:

$$\begin{aligned} p(y;\phi) &= (1-\phi)^{y-1}\phi \\ &= \exp\Big((y-1)\log(1-\phi) + \log\phi\Big) \\ &= \exp\Big((\log(1-\phi))y + \log\phi - \log(1-\phi)\Big) \\ &= \exp\Big((\log(1-\phi))y - \log\frac{1-\phi}{\phi}\Big) \end{aligned}$$

$$\begin{aligned} b(y) &= 1, \\ \eta &= \log(1-\phi), \\ T(y) &= y, \\ \alpha(\eta) &= \log\frac{1-\phi}{\phi}, \\ \phi &= 1-e^{\eta} \end{aligned}$$

ii. [5 points] Suppose that we have an IID training set $\{(x^{(i)}, y^{(i)}), i = 1, ..., m\}$ and we wish to model this using a GLM based on a geometric distribution. Find the log-likelihood $\log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta)$ defined with respect to the entire training set.

Answer: We calculate the log-likelihood for 1 sample as well as for the entire training set:

$$\begin{aligned}
\log p(y^{(i)}|x^{(i)}; \theta) &= \log \Big((1-\phi)^{y^{(i)}-1} \phi \Big) \\
&= (y^{(i)} - 1) \log (1-\phi) + \log \phi \\
&= (\log (1-\phi)) y^{(i)} - \log \frac{1-\phi}{\phi} \\
&= y^{(i)} \log e^{\eta} - \log \frac{e^{\eta}}{1-e^{\eta}} \\
&= \eta y^{(i)} - \eta + \log (1 - e^{\eta}) \\
&= \theta^T x^{(i)} y^{(i)} - \theta^T x^{(i)} + \log (1 - \exp (\theta^T x^{(i)})) \\
&= \theta^T x^{(i)} (y^{(i)} - 1) + \log (1 - \exp (\theta^T x^{(i)}))
\end{aligned}$$

$$\begin{aligned}
l(\theta) &= \log p(y|x; \theta) \\
&= \log \Big(\prod_{i=1}^{m} p(y^{(i)}|x^{(i)}; \theta) \Big) \\
&= \sum_{i=1}^{m} \log \Big(p(y^{(i)}|x^{(i)}; \theta) \Big) \\
&= \sum_{i=1}^{m} \Big(\theta^T x^{(i)} (y^{(i)} - 1) + \log (1 - \exp (\theta^T x^{(i)})) \Big)
\end{aligned}$$

(b) [6 points] Derive the Hessian H and the gradient vector of the log likelihood with respect to θ, and state what one step of Newton's method for maximizing the log likelihood would be.

Answer: To apply Newton's method, we need to find the gradient and Hessian of the log-likelihood:

$$\begin{aligned}
\nabla_\theta l(\theta) &= \nabla_\theta \sum_{i=1}^{m} \Big(\theta^T x^{(i)} y^{(i)} - \theta^T x^{(i)} + \log\left(1 - \exp\left(\theta^T x^{(i)}\right)\right) \Big) \\
&= \sum_{i=1}^{m} \Big(x^{(i)}(y^{(i)} - 1) - \frac{x^{(i)} \exp\left(\theta^T x^{(i)}\right)}{\left(1 - \exp\left(\theta^T x^{(i)}\right)\right)} \Big) \\
&= \sum_{i=1}^{m} \Big(y^{(i)} - \frac{1}{\left(1 - \exp\left(\theta^T x^{(i)}\right)\right)} \Big) x^{(i)} \\
H &= \nabla_\theta \Big(\nabla_\theta (l(\theta))^T \Big) \\
&= -\nabla_\theta \sum_{i=1}^{m} \frac{1}{\left(1 - \exp\left(\theta^T x^{(i)}\right)\right)} x^{(i)T} \\
&= -\sum_{i=1}^{m} \frac{\exp\left(\theta^T x^{(i)}\right)}{\left(1 - \exp\left(\theta^T x^{(i)}\right)\right)^2} x^{(i)} x^{(i)T}
\end{aligned}$$

The Newton's method update rule is then: $\theta := \theta - H^{-1} \nabla_\theta l(\theta)$

(c) [2 points] Show that the Hessian is negative semi-definite. This shows the optimization objective is concave, and hence Newton's method is maximizing log-likelihood.

Answer:

$$\begin{aligned} z^T H z &= -\sum_{i=1}^{m} \frac{\exp\left(\theta^T x^{(i)}\right)}{\left(1-\exp\left(\theta^T x^{(i)}\right)\right)^2} z^T x^{(i)} x^{(i)T} z \\ &= -\sum_{i=1}^{m} \frac{\exp\left(\theta^T x^{(i)}\right)}{\left(1-\exp\left(\theta^T x^{(i)}\right)\right)^2} \|z^T x^{(i)}\|^2 \leq 0 \end{aligned}$$

Therefore, H is negative semidefinite, which means $l(\theta)$ is concave. So we are maximizing it.

4. **[18 points] Support Vector Regression**

   In class, we showed how the SVM can be used for classification. In this problem, we will develop a modified algorithm, called the Support Vector Regression algorithm, which can instead be used for regression, with continuous valued labels $y \in \mathbb{R}$.

   Suppose we are given a training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, where $x^{(i)} \in \mathbb{R}^{(n+1)}$ and $y^{(i)} \in \mathbb{R}$. We would like to find a hypothesis of the form $h_{w,b}(x) = w^T x + b$ with a small value of w. Our (convex) optimization problem is:

$$\begin{array}{lll} \min_{w,b} & \frac{1}{2}\|w\|^2 & \\ \text{s.t.} & y^{(i)} - w^T x^{(i)} - b \leq \epsilon \qquad & i = 1, \ldots, m \quad (1) \\ & w^T x^{(i)} + b - y^{(i)} \leq \epsilon \qquad & i = 1, \ldots, m \quad (2) \end{array}$$

   where $\epsilon > 0$ is a given, fixed value. Notice how the original functional margin constraint has been modified to now represent the distance between the continuous y and our hypothesis' output.

   (a) [4 points] Write down the Lagrangian for the optimization problem above. We suggest you use two sets of Lagrange multipliers α_i and α_i^*, corresponding to the two inequality constraints (labeled (1) and (2) above), so that the Lagrangian would be written $\mathcal{L}(w, b, \alpha, \alpha^*)$.
   Answer:

$$\begin{array}{lll} \min_{w,b} & \frac{1}{2}\|w\|^2 & \\ \text{s.t.} & y^{(i)} - w^T x^{(i)} - b \leq \epsilon \qquad & i = 1, \ldots, m \quad (1) \\ & w^T x^{(i)} + b - y^{(i)} \leq \epsilon \qquad & i = 1, \ldots, m \quad (2) \end{array}$$

   Let $\alpha_i, \alpha_i^* \geq 0$ $(i = 1, \ldots, m)$ be the Lagrange multiplier for (1)-(4) respectively. Then, the Lagrangian can be written as:

$$\begin{aligned} & L(w, b, \alpha, \alpha^*) \\ & = \tfrac{1}{2}\|w\|^2 \\ & - \sum_{i=1}^{m} \alpha_i (\epsilon - y^{(i)} + w^T x^{(i)} + b) \\ & - \sum_{i=1}^{m} \alpha_i^* (\epsilon + y^{(i)} - w^T x^{(i)} - b) \end{aligned}$$

(b) [10 points] Derive the dual optimization problem. You will have to take derivatives of the Lagrangian with respect to w and b.

Answer:

First, the dual objective function can be written as:

$$\theta_D(\alpha, \alpha^*) = \min_{w,b} L(w, b, \alpha, \alpha^*)$$

Now, taking the derivatives of Lagrangian with respect to all primal variables, we have:

$$\partial_w L = w - \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)x^{(i)} = 0$$
$$\partial_b L = \sum_{i=1}^{m}(\alpha_i^* - \alpha_i) = 0$$

Substituting the above two relations back into the Lagrangian, we have:

$$\begin{aligned}
\theta_D(\alpha, \alpha^*) &= \tfrac{1}{2}\|w\|^2 - \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y^{(i)}(\alpha_i - \alpha_i^*) \\
&\quad + b\sum_{i=1}^{m}(\alpha_i^* - \alpha_i) + \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)w^T x^{(i)} \\
\theta_D(\alpha, \alpha^*) &= \tfrac{1}{2}\|w\|^2 - \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y^{(i)}(\alpha_i - \alpha_i^*) + \sum_{i=1}^{m}(\alpha_i^* - \alpha_i)w^T x^{(i)} \\
&= \tfrac{1}{2}\|\sum_{i=1}^{m}(\alpha_i - \alpha_i^*)x^{(i)}\|^2 - \sum_{i=1}^{m}(\alpha_i - \alpha_i^*)\Big(\sum_{j=1}^{m}(\alpha_j - \alpha_j^*)x^{(j)T}x^{(i)}\Big) \\
&\quad - \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y^{(i)}(\alpha_i - \alpha_i^*) \\
&= -\tfrac{1}{2}\sum_{i=1,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x^{(i)T}x^{(j)} - \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y^{(i)}(\alpha_i - \alpha_i^*)
\end{aligned}$$

Now the dual problem can be formulated as:

$$\begin{aligned}
\max_{\alpha_i, \alpha_i^*} \quad & -\tfrac{1}{2}\sum_{i=1,j=1}^{m}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x^{(i)T}x^{(j)} - \epsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{m} y^{(i)}(\alpha_i - \alpha_i^*) \\
\text{s.t.} \quad & \sum_{i=1}^{m}(\alpha_i^* - \alpha_i) = 0 \\
& \alpha_i, \alpha_i^* \geq 0
\end{aligned}$$

[extra space for problem 4 (b)]

(c) [4 points] Show that this algorithm can be kernelized. For this, you have to show that (i) the dual optimization objective can be written in terms of inner-products of training examples; and (ii) at test time, given a new x the hypothesis $h_{w,b}(x)$ can also be computed in terms of inner products.

Answer:

This algorithm can be kernelized because when making prediction at x, we have:

$$f(w,x) = w^T x + b = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) x^{(i)T} x + b = \sum_{i=1}^{m} (\alpha_i - \alpha_i^*) k(x^{(i)}, x) + b$$

This shows that predicting function can be written in a kernel form.

5. **[20 points] Learning Theory**

Suppose you are given a hypothesis $h_0 \in \mathcal{H}$, and your goal is to determine whether h_0 has generalization error within $\eta > 0$ of the best hypothesis, $h^* = \arg\min_{h \in \mathcal{H}} \varepsilon(h)$. More specifically, we say that a hypothesis h is η**-optimal** if $\varepsilon(h) \leq \varepsilon(h^*) + \eta$. Here, we wish to answer the following question:

Given a hypothesis h_0, is h_0 η-optimal?

Let $\delta > 0$ be some fixed constant, and consider a finite hypothesis class $\mathcal{H}$ of size $|\mathcal{H}| = k$. For each $h \in \mathcal{H}$, let $\hat{\varepsilon}(h)$ denote the training error of h with respect to some training set of m IID examples, and let $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{\varepsilon}(h)$ denote the hypothesis that minimizes training error.

Now, consider the following algorithm:

1. Set
$$\gamma := \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}.$$
2. If $\hat{\varepsilon}(h_0) > \hat{\varepsilon}(\hat{h}) + \eta + 2\gamma$, then return NO.
3. If $\hat{\varepsilon}(h_0) < \hat{\varepsilon}(\hat{h}) + \eta - 2\gamma$, then return YES.
4. Otherwise, return UNSURE.

Intuitively, the algorithm works by comparing the training error of h_0 to the training error of the hypothesis $\hat{h}$ with the minimum training error, and returns NO or YES only when $\hat{\varepsilon}(h_0)$ is either significantly larger than or significantly smaller than $\hat{\varepsilon}(\hat{h})+\eta$.

(a) [6 points] First, show that if $\varepsilon(h_0) \leq \varepsilon(h^*) + \eta$ (i.e., h_0 is η-optimal), then the probability that the algorithm returns NO is at most δ.

[extra space for 5 (a)]

Answer:

Suppose that $\varepsilon(h_0) \leq \varepsilon(h^*) + \eta$. Using the Hoeffding inequality, we have that for

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

then with probability at least $1 - \delta$,

$$\begin{aligned}
\hat{\varepsilon}(h_0) &\leq \varepsilon(h_0) + \gamma \\
&\leq \varepsilon(h^*) + \eta + \gamma \\
&\leq \varepsilon(\hat{h}) + \eta + \gamma \\
&\leq \hat{\varepsilon}(\hat{h}) + \eta + 2\gamma.
\end{aligned}$$

Here, the first and last inequalities follow from the fact that under the stated uniform convergence conditions, all hypotheses in $\mathcal{H}$ have empirical errors within γ of their true generalization errors. The second inequality follows from our assumption, and the third inequality follows from the fact that h^* minimizes the true generalization error. Therefore, the reverse condition, $\hat{\varepsilon}(h_0) > \hat{\varepsilon}(\hat{h}) + \eta + 2\gamma$, occurs with probability at most δ.

(b) [6 points] Second, show that if $\varepsilon(h_0) > \varepsilon(h^*) + \eta$ (i.e., h_0 is not η-optimal), then the probability that the algorithm returns YES is at most δ.

Answer:
Suppose that $\varepsilon(h_0) > \varepsilon(h^*) + \eta$. Using the Hoeffding inequality, we have that for

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

then with probability at least $1 - \delta$,

$$\begin{aligned}
\hat{\varepsilon}(h_0) &\geq \varepsilon(h_0) - \gamma \\
&> \varepsilon(h^*) + \eta - \gamma \\
&\geq \hat{\varepsilon}(h^*) + \eta - 2\gamma \\
&\geq \hat{\varepsilon}(\hat{h}) + \eta - 2\gamma.
\end{aligned}$$

Here, the first and third inequalities follow from the fact that under the stated uniform convergence conditions, all hypotheses in $\mathcal{H}$ have empirical errors within γ of their true generalization errors. The second inequality follows from our assumption, and the last inequality follows from the fact that $\hat{h}$ minimizes the empirical error. Therefore, the reverse condition, $\hat{\varepsilon}(h_0) < \hat{\varepsilon}(\hat{h}) + \eta - 2\gamma$ occurs with probability at most δ.

(c) [8 points] Finally, suppose that $h_0 = h^*$, and let $\eta > 0$ and $\delta > 0$ be fixed. Show that if m is sufficiently large, then the probability that the algorithm returns YES is at least $1 - \delta$.

Hint: observe that for fixed η and δ, as $m \to \infty$, we have

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}} \to 0.$$

This means that there are values of m for which $2\gamma < \eta - 2\gamma$.

Answer:
Suppose that $h_0 = h^*$. Using the Hoeffding inequality, we have that for

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}}$$

then with probability at least $1 - \delta$,

$$\begin{aligned} \hat{\varepsilon}(h_0) &\leq \varepsilon(h_0) + \gamma \\ &= \varepsilon(h^*) + \gamma \\ &\leq \varepsilon(\hat{h}) + \gamma \\ &\leq \hat{\varepsilon}(\hat{h}) + 2\gamma. \end{aligned}$$

Here, the first and last inequalities follow from the fact that under the stated uniform convergence conditions, all hypotheses in $\mathcal{H}$ have empirical errors within γ of their true generalization errors. The equality in the second step follows from our assumption, and the inequality in the third step follows from the fact that h^* minimizes the true generalization error. But, observe that for fixed η and δ, as $m \to \infty$, we have

$$\gamma = \sqrt{\frac{1}{2m} \log \frac{2k}{\delta}} \to 0.$$

This implies that for m sufficiently large, $4\gamma < \eta$, or equivalently, $2\gamma < \eta - 2\gamma$. It follows that with probability at least $1 - \delta$, if m is sufficiently large, then

$$\hat{\varepsilon}(h_0) \leq \hat{\varepsilon}(\hat{h}) + \eta - 2\gamma,$$

so the algorithm returns YES.

6. **[24 points] Short answers**

   The following questions require a reasonably short answer (usually at most 2-3 sentences or a figure, though some questions may require longer or shorter explanations).

   To discourage random guessing, one point will be deducted for a wrong answer on true/false or multiple choice questions! Also, no credit will be given for answers without a correct explanation.

   (a) [3 points] You have an implementation of Newton's method and gradient descent. Suppose that one iteration of Newton's method takes twice as long as one iteration of gradient descent. Then, this implies that gradient descent will converge to the optimal objective faster. True/False?

   Answer: False. Newton's method may take fewer steps.

   (b) [3 points] A stochastic gradient descent algorithm for training logistic regression with a fixed learning rate will always converge to exactly the optimal setting of the parameters $\theta^* = \arg\max_\theta \prod_{i=1}^m p(y^{(i)}|x^{(i)};\theta)$, assuming a reasonable choice of the learning rate. True/False?

   Answer: False. A fixed learning rate means that we are always taking a finite step towards improving the log-probability of any single training example in the update equation. Unless the examples are somehow aligned, we will continue jumping from side to side of the optimal solution, and will not be able to get arbitrarily close to it. The learning rate has to approach to zero in the course of the updates for the weights to converge robustly.

(c) [3 points] Given a valid kernel $K(x, y)$ over $\mathbb{R}^m$, is $K_{norm}(x, y) = \frac{K(x,y)}{\sqrt{K(x,x)K(y,y)}}$ a valid kernel?

Answer: Yes. If we write $K(x, y) = \Phi(x)^T\Phi(y)$, then we have:

$$K_{norm}(x, y) = \frac{\Phi(x)^T\Phi(y)}{\sqrt{(\Phi(x)^T\Phi(x))(\Phi(y)^T\Phi(y))}} = \Psi(x)^T\Psi(y)$$

with

$$\Psi(x) = \frac{\Phi(x)}{\sqrt{\Phi(x)^T\Phi(x)}}$$

So K_{norm} (as normalized) is a valid kernel.

(d) [3 points] Consider a 2 class classification problem with a dataset of inputs $\{x^{(1)} = (-1, -1), x^{(2)} = (-1, +1), x^{(3)} = (+1, -1), x^{(4)} = (+1, +1)\}$. Can a linear SVM (with no kernel trick) shatter this set of 4 points?

Answer: No we cannot, since the decision boundary is linear. Let the labels be represented by y. See that we cannot classify in the case $y^{(1)} = +1, y^{(2)} = y^{(3)} = -1, y^{(4)} = +1$.

(e) [3 points] For linear hypotheses (i.e. of the form $h(x) = w^T x + b$), the vector of learned weights w is always perpendicular to the separating hyperplane. True/False? Provide a counterexample if False, or a brief explanation if True.

Answer: True. For a linear separating boundary, the hyperplane is defined by the set $\{x|w^T x = -b\}$. The inner product $w^T x$ geometrically represents the projection of x onto w. The set of all points whose projection onto w is constant ($-b$) forms a line that must be perpendicular to w. So h is perpendicular to w. This fact is necessary in the formulation of geometric margins for the linear SVM.

(f) [3 points] Let $\mathcal{H}$ be a set of classifiers with a VC dimension of 5. Consider a set of 5 training examples $\{(x^{(1)}, y^{(1)}), ..., (x^{(5)}, y^{(5)})\}$. Now we select a classifier h^* from $\mathcal{H}$ by minimizing the classification error on the training set. Which one of the following is true?

i. $x^{(5)}$ will certainly be classified correctly (i.e. $h^*(x^{(5)}) = y^{(5)}$)
ii. $x^{(5)}$ will certainly be classified incorrectly (i.e. $h^*(x^{(5)}) \neq y^{(5)}$)
iii. We cannot tell

Briefly justify your answer.

Answer: We cannot tell. Since the VC-dimension of $\mathcal{H}$ is 5, $\mathcal{H}$ can shatter **some** set of points of size five. These points could be $\{x^{(1)}, \ldots, x^{(5)}\}$ but do not necessarily have to be. As a result, there is no guarantee that the given set will be shattered and so we can make no claims about the classification of $x^{(5)}$.

(g) [6 points] Suppose you would like to use a linear regression model in order to predict the price of houses. In your model, you use the features $x_0 = 1$, $x_1 =$ size in square meters, $x_2 =$ height of roof in meters. Now, suppose a friend repeats the same analysis using exactly the same training set, only he represents the data instead using features $x'_0 = 1$, $x'_1 = x_1$, and $x'_2 =$ height in cm (so $x'_2 = 100x_2$).

i. [3 points] Suppose both of you run linear regression, solving for the parameters via the Normal equations. (Assume there are no degeneracies, so this gives a unique solution to the parameters.) You get parameters θ_0, θ_1, θ_2; your friend gets θ'_0, θ'_1, θ'_2. Then $\theta'_0 = \theta_0$, $\theta'_1 = \theta_1$, $\theta'_2 = \frac{1}{100}\theta_2$. True/False?

Answer: True. Observe that running a single step of Newton's method, for a linear regression problem, is equivalent to solving the Normal equations. The result then follows from the invariance of Newton's method to linear reparameterizations.

ii. [3 points] Suppose both of you run linear regression, initializing the parameters to 0, and compare your results after running just *one* iteration of batch gradient descent. You get parameters θ_0, θ_1, θ_2; your friend gets θ'_0, θ'_1, θ'_2. Then $\theta'_0 = \theta_0$, $\theta'_1 = \theta_1$, $\theta'_2 = \frac{1}{100}\theta_2$. True/False?

Answer: False. Recall that gradient descent is not invariant to linear reparameterizations.