CS 229, Autumn 2016
Problem Set #2: Naive Bayes, SVMs, and Theory

Due Wednesday, November 2 at 11:00 am on Gradescope.

Notes: (1) These questions require thought, but do not require long answers. Please be as concise as possible. (2) If you have a question about this homework, we encourage you to post your question on our Piazza forum, at `https://piazza.com/stanford/autumn2016/cs229`. (3) If you missed the first lecture or are unfamiliar with the collaboration or honor code policy, please read the policy on Handout #1 (available from the course website) before starting work. (4) For problems that require programming, please include in your submission a printout of your code (with comments) and any figures that you are asked to plot.

If you are scanning your document by cellphone, please check the Piazza forum for recommended cellphone scanning apps and best practices.

1. **[15 points] Constructing kernels**

   In class, we saw that by choosing a kernel $K(x, z) = \phi(x)^T \phi(z)$, we can implicitly map data to a high dimensional space, and have the SVM algorithm work in that space. One way to generate kernels is to explicitly define the mapping ϕ to a higher dimensional space, and then work out the corresponding K.

   However in this question we are interested in direct construction of kernels. I.e., suppose we have a function $K(x, z)$ that we think gives an appropriate similarity measure for our learning problem, and we are considering plugging K into the SVM as the kernel function. However for $K(x, z)$ to be a valid kernel, it must correspond to an inner product in some higher dimensional space resulting from some feature mapping ϕ. Mercer's theorem tells us that $K(x, z)$ is a (Mercer) kernel if and only if for any finite set $\{x^{(1)}, \ldots, x^{(m)}\}$, the matrix K is symmetric and positive semidefinite, where the square matrix $K \in \mathbb{R}^{m \times m}$ is given by $K_{ij} = K(x^{(i)}, x^{(j)})$.

   Now here comes the question: Let K_1, K_2 be kernels over $\mathbb{R}^n \times \mathbb{R}^n$, let $a \in \mathbb{R}^+$ be a positive real number, let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a real-valued function, let $\phi : \mathbb{R}^n \to \mathbb{R}^d$ be a function mapping from $\mathbb{R}^n$ to $\mathbb{R}^d$, let K_3 be a kernel over $\mathbb{R}^d \times \mathbb{R}^d$, and let $p(x)$ a polynomial over x with *positive* coefficients.

   For each of the functions K below, state whether it is necessarily a kernel. If you think it is, prove it; if you think it isn't, give a counter-example.

   (a) [1 points] $K(x, z) = K_1(x, z) + K_2(x, z)$

   (b) [1 points] $K(x, z) = K_1(x, z) - K_2(x, z)$

   (c) [1 points] $K(x, z) = aK_1(x, z)$

   (d) [1 points] $K(x, z) = -aK_1(x, z)$

   (e) [5 points] $K(x, z) = K_1(x, z)K_2(x, z)$

   (f) [2 points] $K(x, z) = f(x)f(z)$

   (g) [2 points] $K(x, z) = K_3(\phi(x), \phi(z))$

   (h) [2 points] $K(x, z) = p(K_1(x, z))$

[Hint: For part (e), the answer is that the K there *is* indeed a kernel. You still have to prove it, though. (This one may be harder than the rest.) This result may also be useful for another part of the problem.]

2. **[10 points] Kernelizing the Perceptron**

Let there be a binary classification problem with $y \in \{-1, 1\}$. The perceptron uses hypotheses of the form $h_\theta(x) = g(\theta^T x)$, where $g(z) = \text{sign}(z) = 1$ if $z \geq 0$, -1 otherwise. In this problem we will consider a stochastic gradient descent-like implementation of the perceptron algorithm where each update to the parameters θ is made using only one training example. However, unlike stochastic gradient descent, the perceptron algorithm will only make one pass through the entire training set. The update rule for this version of the perceptron algorithm is given by

$$\theta^{(i+1)} := \begin{cases} \theta^{(i)} + \alpha y^{(i+1)} x^{(i+1)} & \text{if } h_{\theta^{(i)}}(x^{(i+1)}) y^{(i+1)} < 0 \\ \theta^{(i)} & \text{otherwise,} \end{cases}$$

where $\theta^{(i)}$ is the value of the parameters after the algorithm has seen the first i training examples. Prior to seeing any training examples, $\theta^{(0)}$ is initialized to $\vec{0}$.

Let K be a Mercer kernel corresponding to some very high-dimensional feature mapping ϕ. Suppose ϕ is so high-dimensional (say, ∞-dimensional) that it's infeasible to ever represent $\phi(x)$ explicitly. Describe how you would apply the "kernel trick" to the perceptron to make it work in the high-dimensional feature space ϕ, but without ever explicitly computing $\phi(x)$. [Note: You don't have to worry about the intercept term. If you like, think of ϕ as having the property that $\phi_0(x) = 1$ so that this is taken care of.] Your description should specify

(a) How you will (implicitly) represent the high-dimensional parameter vector $\theta^{(i)}$, including how the initial value $\theta^{(0)} = \vec{0}$ is represented (note that $\theta^{(i)}$ is now a vector whose dimension is the same as the feature vectors $\phi(x)$);

(b) How you will efficiently make a prediction on a new input $x^{(i+1)}$. I.e., how you will compute $h_{\theta^{(i)}}(x^{(i+1)}) = g({\theta^{(i)}}^T \phi(x^{(i+1)}))$, using your representation of $\theta^{(i)}$; and

(c) How you will modify the update rule given above to perform an update to θ on a new training example $(x^{(i+1)}, y^{(i+1)})$; i.e., using the update rule corresponding to the feature mapping ϕ:

$$\theta^{(i+1)} := \theta^{(i)} + \alpha \mathbf{1}\{{\theta^{(i)}}^T \phi(x^{(i+1)}) y^{(i+1)} < 0\} y^{(i+1)} \phi(x^{(i+1)}).$$

[Hint: our discussion of the representer theorem may be useful.]

4. **[20 points] Properties of VC dimension**

   In this problem, we investigate a few properties of the Vapnik-Chervonenkis dimension, mostly relating to how VC(H) increases as the set H increases. For each part of this problem, you should state whether the given statement is true, and justify your answer with either a formal proof or a counter-example.

   (a) Let two hypothesis classes H_1 and H_2 satisfy $H_1 \subseteq H_2$. Prove or disprove: $\text{VC}(H_1) \leq \text{VC}(H_2)$.

   (b) Let $H_1 = H_2 \cup \{h_1, \ldots, h_k\}$. (I.e., H_1 is the union of H_2 and some set of k additional hypotheses.) Prove or disprove: $\text{VC}(H_1) \leq \text{VC}(H_2) + k$. [Hint: You might want to start by considering the case of $k = 1$.]

   (c) Let $H_1 = H_2 \cup H_3$. Prove or disprove: $\text{VC}(H_1) \leq \text{VC}(H_2) + \text{VC}(H_3)$.

5. **[20 points] Training and testing on different distributions**

   In the discussion in class about learning theory, a key assumption was that we trained and tested our learning algorithms on the same distribution $\mathcal{D}$. In this problem, we'll investigate one special case of training and testing on different distributions. Specifically, we will consider what happens when the training labels are *noisy*, but the test labels are not.

   Consider a binary classification problem with labels $y \in \{0, 1\}$, and let $\mathcal{D}$ be a distribution over (x, y), that we'll think of as the original, "clean" or "uncorrupted" distribution. Define $\mathcal{D}_\tau$ to be a "corrupted" distribution over (x, y) which is the same as $\mathcal{D}$, except that the labels y have some probability $0 \leq \tau < 0.5$ of being flipped. Thus, to sample from $\mathcal{D}_\tau$, we would first sample (x, y) from $\mathcal{D}$, and then with probability τ (independently of the observed x and y) replace y with $1 - y$. Note that $\mathcal{D}_0 = \mathcal{D}$.

   The distribution $\mathcal{D}_\tau$ models a setting in which an unreliable human (or other source) is labeling your training data for you, and on each example he/she has a probability τ of mislabeling it. Even though our training data is corrupted, we are still interested in evaluating our hypotheses with respect to the original, uncorrupted distribution $\mathcal{D}$.

   We define the generalization error *with respect to* $\mathcal{D}_\tau$ to be

   $$\varepsilon_\tau(h) = P_{(x,y)\sim\mathcal{D}_\tau}[h(x) \neq y].$$

   Note that $\varepsilon_0(h)$ is the generalization error with respect to the "clean" distribution; it is with respect to ε_0 that we wish to evaluate our hypotheses.

(a) For any hypothesis h, the quantity $\varepsilon_0(h)$ can be calculated as a function of $\varepsilon_\tau(h)$ and τ. Write down a formula for $\varepsilon_0(h)$ in terms of $\varepsilon_\tau(h)$ and τ, and justify your answer.

(b) Let $|H|$ be finite, and suppose our training set $S = \{(x^{(i)}, y^{(i)}); i = 1, \ldots, m\}$ is obtained by drawing m examples IID from the corrupted distribution $\mathcal{D}_\tau$. Suppose we pick $h \in H$ using empirical risk minimization: $\hat{h} = \arg\min_{h \in H} \hat{\varepsilon}_S(h)$. Also, let $h^* = \arg\min_{h \in H} \varepsilon_0(h)$.

Let any $\delta, \gamma > 0$ be given. Prove that for

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

to hold with probability $1 - \delta$, it suffices that

$$m \geq \frac{1}{2(1-2\tau)^2\gamma^2} \log \frac{2|H|}{\delta}.$$

Remark. This result suggests that, roughly, m examples that have been corrupted at noise level τ are worth about as much as $(1-2\tau)^2 m$ uncorrupted training examples. This is a useful rule-of-thumb to know if you ever need to decide whether/how much to pay for a more reliable source of training data. (If you've taken a class in information theory, you may also have heard that $(1-\mathcal{H}(\tau))m$ is a good estimate of the information in the m corrupted examples, where $\mathcal{H}(\tau) = -(\tau \log_2 \tau + (1-\tau)\log_2(1-\tau))$ is the "binary entropy" function. And indeed, the functions $(1-2\tau)^2$ and $1-\mathcal{H}(\tau)$ are quite close to each other.)

(c) Comment **briefly** on what happens as τ approaches 0.5.