ECE 543 STATISTICAL LEARNING THEORY **SPRING 2019**
PROBLEM SET 4 **Due beginning of class, Tuesday, March 26**

4. Regression with quadratic risk, stability of learning algorithms

Assigned reading: Chapters 9, Sections 10.1-10.4, and Sections 3.4-3.5 of course notes.
Optional supplementary reading: Shalev-Shwartz and Ben-David, *Understanding Machine Learning, from Theory to Algorithms,* Chapters 14-16.

(Problems and solutions)

1. **[Derivation of AdaBoost]**
Let $\mathcal{G}$ be a set of base classifiers mapping X to the label set $\{-1,1\}$. Focus on a classifier of the form $\mathsf{sgn}(f)$ where $f(x)=\sum_{k=1}^{t}\alpha_k g_k(x)$ for some $t\geq 1$, $g_1,\ldots,g_t\in\mathcal{G}$, and $\alpha_1,\ldots,\alpha_t\in\mathbb{R}$. The 0-1 loss on a sample (x,y) is $\ell(y,f(x))=\mathbf{1}_{\{y\neq \mathsf{sgn} f(x)\}}$ and the surrogate loss using $\varphi(x)=\mathrm{e}^x$ is given by $\ell_\phi(x,y)=\exp(-yf(x))$.

Let $z^n=((x_1,y_1),\ldots,(x_n,y_n))$ be a set of labeled data points, and suppose $w^n=(w_1,\ldots,w_n)$ is a vector of positive weights for the n data points. Then (z^n,w^n) represents a *weighted* sample set. Suppose there is a learning algorithm WL (for "weak learner") that maps weighted sample sets to base classifiers: $WL:\mathsf{Z}^n\times(0,\infty)^n\to\mathcal{G}$. Specifically, suppose the weak learner is a weighted ERM algorithm for the base class $\mathcal{G}$ and 0-1 loss:

$$WL(z^n,w^n)=\arg\min_{g\in\mathcal{G}}\frac{1}{n}\sum_{i=1}^{n}w_i\mathbf{1}_{\{y_i\neq g(x_i)\}}. \tag{1}$$

(a) Show that (1) is equivalent to the following, where α is an arbitrary positive constant:

$$WL(z^n,w^n)=\arg\min_{g\in\mathcal{G}}\frac{1}{n}\sum_{i=1}^{n}w_i\exp(-y_i\alpha g(x_i)).$$

(Hint: Remember that functions in $\mathcal{G}$ are $\{-1,1\}$–valued.)
Solution: Since both y_i and $g(x_i)$ take values in $\{-1,1\}$.

$$\exp(-y_i\alpha g(x_i))=\mathrm{e}^{-\alpha}+(\mathrm{e}^{\alpha}-\mathrm{e}^{-\alpha})\mathbf{1}_{\{y_i\neq g(x_i)\}}.$$

So $\frac{1}{n}\sum_{i=1}^{n}w_i\exp(-y_i\alpha g(x_i))$ is an affine function with positive slope, of $\frac{1}{n}\sum_{i=1}^{n}w_i\mathbf{1}_{\{y_i\neq g(x_i)\}}$. Therefore, minimizing one of them with respect to g is equivalent to minimizing the other.

(b) Suppose (X,Y) represents a random labeled sample with values in $\mathsf{X}\times\{-1,1\}$, and suppose g is a classifier; $g:\mathsf{X}\to\{-1,1\}$. Find the α that minimizes $\mathbb{E}\left[\mathrm{e}^{-Y(\alpha g(X))}\right]$, the expected surrogate loss for αg. Express both the optimal α and the resulting value of the minimum in terms of ϵ, where $\epsilon=\mathbb{P}\{Y\neq g(X)\}$.
Solution: $\mathbb{E}\left[\mathrm{e}^{-Y(\alpha g(X))}\right]=\epsilon\mathrm{e}^{-\alpha}+(1-\epsilon)\mathrm{e}^{\alpha}$. To find the minimizing value we set the derivative with respect to α to zero, yielding $\alpha^*=\frac{1}{2}\log\left(\frac{1}{\epsilon}-1\right)$ and value $\epsilon\mathrm{e}^{-\alpha^*}+(1-\epsilon)\mathrm{e}^{\alpha^*}=2\sqrt{\epsilon(1-\epsilon)}$.

(c) For f described above, the empirical surrogate risk is given by:

$$A_{n,\varphi}(f)=\frac{1}{n}\sum_{i=1}^{n}\exp\left(-y_i\sum_{k=1}^{t}\alpha_k g_k(x_i)\right).$$

Given $\alpha_1,\ldots,\alpha_{t-1}$ and $g_1,\ldots,g_{t-1}$, show that selecting α_t and g_t to minimize $A_{n,\varphi}(f)$ can be decomposed into two steps:

Step one: Find $g_t\in\mathcal{G}$ to minimize the weighted empirical probability of error, ϵ_t, using weights

$$w_i^{(t)}=\exp\left(-y_i\sum_{k=1}^{t-1}\alpha_k g_k(x_i)\right). \tag{2}$$

(For $t=1$, $w_i^{(1)}=1$.)

Step two: Find α_t.

In particular, explain why the weights $w_1^{(t)}, \ldots, w_n^{(t)}$ given in (2) are appropriate for the first step, and then describe the choice of α_t. (You can assume $\alpha_t > 0$, which will be true if $\epsilon_t < 1/2$, meaning that g_t does better than random guessing.)

Solution: The empirical surrogate loss for f can be rewritten as

$$\begin{aligned} A_{n,\varphi}(f) &= \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i \sum_{k=1}^{t-1} \alpha_k g_k(x_i)\right) \exp\left(-y_i \alpha_t g_t(x_i)\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} w_i^{(t)} \exp\left(-y_i \alpha_t g_t(x_i)\right). \end{aligned} \tag{3}$$

By part (a) above, minimizing $A_{n,\varphi}(f)$ as represented in (3) with respect to g_t is equivalent to minimizing the empirical weighted 0-1 loss ϵ_t, given by

$$\epsilon_t = \sum_{i=1}^{n} \frac{w_i^{(t)}}{\sum_{j=1}^{n} w_j^{(t)}} \mathbf{1}_{\{y_i \neq g(x_i)\}}.$$

Once g_t is found and the corresponding value ϵ_t is determined, part (b) above applied to the weighted empirical distribution yields that the choice for α_t that minimizes the right-hand side of (3) is given by $\alpha_t = \frac{1}{2}\log\left(\frac{1}{\epsilon_t} - 1\right)$.

(d) Let $f^{(t)}$ denote the classifier f constructed by the algorithm after t terms have been found for f. Show that $A_{n,\varphi}(f^{(t)}) = A_{n,\varphi}(f^{(t-1)}) 2\sqrt{\epsilon_t(1-\epsilon_t)}$ for $t \geq 1$.

Solution: With the choice of $w_i^{(t)}$ above

$$\begin{aligned} A_{n,\varphi}(f^{(t)}) &= \frac{1}{n}\sum_{i=1}^{n} w_i^{(t+1)} \\ &= A_{n,\varphi}(f^{(t-1)}) \min_{\alpha} \sum_{i=1}^{n} \frac{w_i^{(t)}}{\sum_{j=1}^{n} w_j^{(t)}} \exp(-y_i \alpha g_t(x_i)) \\ &= A_{n,\varphi}(f^{(t-1)}) 2\sqrt{\epsilon_t(1-\epsilon_t)}, \end{aligned}$$

where for the last inequality we applied part (b) for the empirical distribution with weight vector $w^{(t)}$.

2. **[On necessity of UCEM for concept learning]**
It is stated in Section 10.1 that for the problem of binary concept classification, the UCEM property is necessary for learnability. Why?

Solution: By the fundamental theorem of concept learning, if a problem is PAC learnable then $V(\mathcal{C}) < \infty$. If $V(\mathcal{C}) < \infty$ then the proof of consistency (i.e. PAC learnability) we've seen is through the UCEM property, based on McDiarmid's inequality, Sauer-Shelah lemma, the symmetrization trick, and the finite-class lemma for Rademacher averages.

3. **[UCEM, generalization, and consistency]**
Consider the learning problem $(\mathsf{Z}, \mathcal{P}, \mathcal{F}, \ell)$ described in Section 10.1. Recall that an algorithm A :

$$\begin{aligned} \text{generalizes if } g_n(A) &:= \sup_P \mathbf{E}_P \left|L(A(Z^n)) - L_n(A(Z^n))\right| \xrightarrow{n\to\infty} 0, \\ \text{is consistent if } c_n(A) &:= \sup_P \mathbf{E}_P \left[L(A(Z^n)) - L^*\right] \xrightarrow{n\to\infty} 0, \\ \text{is AERM if } e_n(A) &:= \sup_P \mathbf{E}_P \left[L_n(A(Z^n)) - L_n^*\right] \xrightarrow{n\to\infty} 0. \end{aligned}$$

(a) Is the following true or false? If there exists an AERM algorithm A that is consistent, then any other AERM algorithm A' is also consistent. Explain.

Solution: No. Here is a counter example. Consider the classification problem $(\mathsf{X}, \mathsf{Y} = \{0,1\}, \mathcal{C}, \mathcal{P})$ such that $\mathsf{X} = [0,1]$, $\mathcal{C}$ is the set of sets C such that C is the union of a finite number of intervals in $[0,1]$, and $\mathcal{P} = \{P\}$, where P is the distribution of (X,Y) such that X is uniformly distributed over $[0,1]$ and $Y \equiv 1$. Let A denote the algorithm such that $A(z^n) = [0,1]$ for any z^n. Then A is an ERM algorithm (it has zero loss for any data set produced by independent draws from P) and it is consistent (it's output has zero loss on fresh samples). Let $\widetilde{A}$ be any ERM algorithm such that the sum of the lengths of the intervals in the output $C = \widetilde{A}(x^n)$ is less than or equal to $1/2$ for any choice of z^n. Such an algorithm exists–the ERM property simply means that the output set should contain all the x_i's of the data. The expected loss for any output of $\widetilde{A}$ applied on fresh samples is at least $1/2$ for all n. Thus, $\widetilde{A}$ is not consistent. In summary, both A and $\widetilde{A}$ are ERM, but only A is consistent.

(b) Suppose the learning problem itself has the uniform convergence of empirical moments (UCEM) property as defined in Chapter 5:

$$q(n,\epsilon) := \sup_{P\in\mathcal{P}} P^n \left(Z^n \in \mathsf{Z}^n : \sup_{f\in\mathcal{F}} |P_n(\ell_f) - P(\ell_f)| \geq \epsilon \right) \xrightarrow{n\to\infty} 0. \tag{4}$$

and suppose the range of ℓ is a bounded interval $[0,B]$. Show that any learning algorithm A generalizes. (Hint: Bound $g_n(A)$ in terms of ϵ, $q(n,\epsilon)$ and B and then let $\epsilon \to 0$ as $n \to \infty$ in a suitable way.)

Solution: Since $A(Z^n)$ takes values in $\mathcal{F}$,

$$P\left[|L(A(Z^n)) - L_n(A(Z^n))| \geq \epsilon\right] \leq P^n \left(\sup_{f\in\mathcal{F}} |P_n(\ell_f) - P(\ell_f)| \geq \epsilon \right) = q(n,\epsilon)$$

Thus, for any n, ϵ,

$$g_n(A) := \sup_P \mathbf{E}_P \left|L(A(Z^n)) - L_n(A(Z^n))\right| \leq \epsilon + Bq(n,\epsilon)$$

By assumption, $q(n,\epsilon) \xrightarrow{n\to\infty} 0$ for any $\epsilon > 0$. It follows there is a sequence $(\epsilon_n)_{n\geq 1}$ such $\epsilon_n \xrightarrow{n\to\infty} 0$ and $q(n,\epsilon_n) \xrightarrow{n\to\infty} 0$. Thus, $g_n(A) \leq \epsilon_n + Bq(n,\epsilon_n) \xrightarrow{n\to\infty} 0$.

4. **[On smoothness and strong convexity]**

(a) Suppose $f : \mathbb{R}^d \to \mathbb{R}$ is defined by $f(x) = \|x\|^2 + 2\|x\|$. Determine whether f is m- strongly convex for some $m > 0$, and if so, determine the largest such m. And determine whether f is M-smooth for some finite M, and if so, determine the smallest such M. (Hint: Consider the case $d = 1$ first.)

Solution: f is m-strongly convex for $m = 2$ and not for any larger values of m, because $f(x) - \frac{m}{2}\|x\|^2$ is convex for $m = 2$ but not for $m > 2$ (for the later, consider f restricted to a multiple of a nonzero vector in $\mathbb{R}^2$). f is not M smooth for any $M \geq 0$ because it is not differentiable at zero.

(b) Repeat part (a) for the function defined by $g(x) = ((\|x\| - 1)_+)^2$.

Solution: g is not m-strongly convex for any $m > 0$ because $g \equiv 0$ in a neighborhood of 0. g is M smooth for $M = 2$, by part (c).

(c) Suppose $h : [0, +\infty) \to \mathbb{R}$ is an m strongly convex, M smooth function for some constants $0 < m \leq M < \infty$ such that $h(0) = h'(0) = 0$. Suppose $H : \mathbb{R}^d \to \mathbb{R}$ is defined by $H(x) = h(\|x\|)$. Repeat part (a) for the function H. You may assume that h is twice continuously differentiable. Hint: One approach for smoothness is to begin by finding the gradient of H, and, for nonzero vectors x, the Hessian $\nabla^2 H$.

Solution: The function H is m-strongly convex. Here is a proof. Since h is m-strongly convex, the function k defined by $k(x) = h(x) - \frac{m}{2}x^2$ is a convex function on $\mathbb{R}$. Also, $k'(0) = h'(0) = 0$

so that $k(x)$ is increasing in x over $0 \le x < \infty$. Therefore, by convexity of $\|x\|$ and the convexity and monotonicity of k, $k(\|x\|)$ is a convex function. In other words, $x \mapsto h(\|x\|) - \frac{m}{2}\|x\|^2$ is a convex function. Therefore, h is m-strongly convex. It is not necessarily strongly convex for a larger constant as can seen by looking along multiples of a single vector.

The function H is M-smooth. Here is a proof. By the chain rule we find that H is continuously differentiable with

$$\nabla H(x) = \begin{cases} \frac{h'(\|x\|)}{\|x\|} x & x \neq 0 \\ 0 & x = 0. \end{cases}$$

and the Hessian of H is given for $x \neq 0$ by

$$\begin{aligned} \nabla^2 H(x) &= \left[h''(\|x\|) - \frac{h'(\|x\|)}{\|x\|} \right] vv^* + \frac{h'(\|x\|)}{\|x\|} I \\ &\prec \left[M - \frac{h'(\|x\|)}{\|x\|} \right] vv^* + \frac{h'(\|x\|)}{\|x\|} I, \end{aligned}$$

where $v = \frac{x}{\|x\|}$ and I is the identity matrix. Since v is a unit length vector, $vv^* \prec I$, and the M-smoothness of h together with $h'(0) = 0$ implies that $0 \le \frac{h'(\|x\|)}{\|x\|} \le M$. Thus $\nabla^2 H(x) \prec \left[M - \frac{h'(\|x\|)}{\|x\|} + \frac{h'(\|x\|)}{\|x\|} \right] I = MI$. Therefore, for any x, x'' such that the line segment x, x' does not go through the origin, $|\nabla H(x) - \nabla H(x')| \le M\|x - x'\|$. By continuity of ∇H, this holds for all x, x', which by definition means H is M-smooth.

5. **[Stochastic gradient descent for solving soft SVM and the kernel trick]**
Consider the regularized ERM problem for half-space classifiers and surrogate loss using the hinge penalty function, $\varphi(x) = (1 + x)_+$:

$$\min_w \left(\frac{\tau \|w\|^2}{2} + \frac{1}{n} \sum_{i=1}^n (1 - y_i(\langle w, x_i \rangle))_+ \right) \quad \text{with respect to } w \in \mathbb{R}^d \tag{5}$$

where the labeled data is $((x_1, y_1), \ldots, (x_n, y_n)) \in (\mathbb{R}^d \times \{\pm 1\})^n$. (We could have used $\langle w, x_i \rangle + b$ instead of $\langle w, x_i \rangle$ and $\|w\|^2 + b^2$ instead of $\|w\|^2$, but by adding a first coordinate equal to 1 to each x_i, the b can be thought of as the first coordinate of w.) Let $F(w)$ denote the objective function, defined to be the quantity in parentheses in (5).

(a) Show that a subgradient of the objective function F at a point $w \in \mathbb{R}^d$ is given by

$$\tau w - \frac{1}{n} \sum_{i=1}^n y_i x_i \mathbf{1}_{\{y_i \langle w, x_i \rangle < 1\}}.$$

Solution: The convex function $\|w\|^2/2$ is differentiable with gradient w.

We claim that if $\varphi : \mathbb{R} \to \mathbb{R}$ is a convex function with subgradient g, and if $L : \mathbb{R}^d \to \mathbb{R}$ is a linear function, then $\varphi \circ L$ has subgradient $(g \circ L)\nabla L$, where $\circ$ denotes composition of functions. To verify the claim, by definition of g, for any u, $\varphi(u') - \varphi(u) \ge g(u)(u' - u)$. Therefore, if $w, w' \in \mathbb{R}^d$,

$$\varphi \circ L(w') - \varphi \circ L(w) \ge (g \circ L(w))(L(w') - L(w)) = \langle (g \circ L(w)) \nabla L, w' - w \rangle,$$

which, by definition, verifies the claim.

By the reasoning of the first paragrpah abouve, $-\mathbf{1}_{\{u<1\}}$ is a subgradient of $(1-u)_+$ for all u. The linear function $w \to y_i \langle w, x_i \rangle$ has gradient $y_i x_i$. So by the claim just proved, $(1 - y_i \langle w, x_i \rangle))_+$ has subgradient $-y_i x_i \mathbf{1}_{\{y_i \langle w, x_i \rangle < 1\}}$. Since the sum of the subgradients of a finite number of convex functions is a subgradient of the sum of the functions, the result follows.

(b) Hence, an iteration of the gradient descent algorithm takes the form

$$w^{(t+1)} = w^{(t)} + \eta_t \left(-\tau w^{(t)} + \frac{1}{n} \sum_{i=1}^{n} y_i x_i \mathbf{1}_{\{y_i \langle w^{(t)}, x_i \rangle < 1\}} \right), \tag{6}$$

where η_t is a step size multiplier with $\eta_t > 0$. The idea of *stochastic gradient descent* (SGD) is to replace the subgradient by a random vector v_t such that $E[v_t]$ is a subgradient. A popular way to select such a v_t is to replace the average over n terms appearing in (6) by one term selected uniformly at random. Hence, an iteration of the SGD algorithm takes the form

$$w^{(t+1)} = w^{(t)} + \eta_t \left(-\tau w^{(t)} + y_{i_t} x_{i_t} \mathbf{1}_{\{y_{i_t} \langle w^{(t)}, x_{i_t} \rangle < 1\}} \right), \tag{7}$$

where i_t is selected uniformly at random from $[n]$. The initial vector could be taken to be $w^{(0)} = 0$. An intuitive interpretation of (7) is that unless the sample (x_{i_t}, y_{i_t}) is correctly labeled and far enough away from the decision boundary, w is moved a bit in the direction $y_{i_t} x_{i_t}$ to increase $y_{i_t} \langle w, x_{i_t} \rangle$. More elaborate versions of SGD have been devised with performance guarantees. They would use a specific choice of stepsize multipliers (η_t) and may return an average of computed values instead of a final value. However, for now, we stick to the basic algorithm here.

The idea of the *kernel trick* in machine learning is that if there is an algorithm that uses feature vectors only through their inner products, then the algorithm can be implemented in $\mathbb{R}^n$, where n is the number of sample points, indpendently of the dimension of the feature space X (i.e. d in this case). To illustrate that, show how the iteration (7), which deals with vectors of length d, can be implemented using vectors of length n. (Hint: Note that, by induction, $w^{(t)}$ is always in the span of the n vectors $(x_i)_{i \in [n]}$. For the particular kernel $K(x, x') = \langle x, x' \rangle$, the function $K_x(\cdot) = \langle \cdot, x \rangle$, so the association between K_x and x is particularly simple.)

Solution: Rather than computing $w^{(t)}$, we can compute $c^{(t)} \in \mathbb{R}^n$, which is a vector such that $w^{(t)} = \sum_{j=1}^{n} c_j^{(t)} x_j$. Since

$$\langle w^{(t)}, x_{i_t} \rangle = \left\langle \sum_{j=1}^{n} c_j^{(t)} x_j, x_{i_t} \right\rangle = \sum_{j=1}^{n} c_j^{(t)} \langle x_j, x_{i_t} \rangle = \sum_{j=1}^{n} c_j^{(t)} K(x_j, x_{i_t}),$$

the iteration (7) of the SGD algorithm can be computed using:

$$c^{(t+1)} = c^{(t)} + \eta_t \left(-\tau c^{(t)} + y_{i_t} \delta_{i_t} \mathbf{1}_{\{y_{i_t} \sum_{j=1}^{n} c_j^{(t)} K(x_j, x_{i_t}) < 1\}} \right), \tag{8}$$

where δ_i denotes the i^{th} standard basis vector in $\mathbb{R}^n$ (i.e. it has i^{th} coordinate equal to one and all others equal to zero). The initial vector would be $c^{(0)} = 0$.