CS489/698: Introduction to Machine Learning

Homework 2

Due: 11:59 pm, October 12, 2017, submit on LEARN.

Include your name, student number and session!

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TAs can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

Exercise 1: k-nearest neighbors (30 pts)

Convention: The Matlab notation $X_{:,j}$ means the j-th column of X while $X_{i,:}$ means the i-th row of X.

Algorithm 1: k-nearest neighbors.

Input: $X \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{0,1,\dots,9\}^n$ (training set), $Z \in \mathbb{R}^{m\times d}$ (test set), distance metric $\mathsf{d}(\cdot,\cdot)$, $k \in \mathbb{N}$
Output: $\hat{\mathbf{y}} \in \{0,1,\dots,9\}^m$

1 **for** $t = 1,2,\dots,m$ **do** // for each test sample
2 **for** $i = 1,2,\dots,n$ **do** // for each training sample
3 compute $D_{i,t} = \mathsf{d}(X_{i,:}, Z_{t,:})$
4 find k smallest entries in $D_{:,t}$, indexed by $t_1,\dots,t_k$ // break ties arbitrarily
5 $\hat{y}_t \leftarrow$ majority vote of $y_{t_1},\dots,y_{t_k}$ // break ties arbitrarily

Implement the k-nearest neighbor algorithm in Algorithm 1, and test it on the MNIST dataset (also available on course website). Specify the distance metric $\mathsf{d}(\cdot,\cdot)$ that you use. To select k, use 10-fold cross-validation but refrain from peeking at the test set. [A naive implementation would be too slow and the memory consumption would be huge. Try to break up the computation and recycle. For Matlab (and maybe python as well), avoid unnecessary for-loops by using matrix-matrix multiplication.] [If it is still too slow to run cross-validation, use a portion of the training set to do the job.] Plot the averaged validation error (y-axis) against k (x-axis), with the "best" k highlighted. After k is selected, report the error rate on the test set. [One could also cross-validate the distance metric if there are a few.]

An $O(n \log n)$ or $O(nk)$ implementation for line 4 will receive full credit but you should try to strike for $O(n)$ time complexity. [Some algorithm from CS341 might be useful.]

[Honesty rule: In your experiment, you can only inspect the test set once—when you actually compute the test error. If you want to try some ideas to improve performance or tune k, split the training set to do so but never peek the test set, please!]

Bonus (5 extra pts). Any implementation that respects the honesty rule and achieves lower than .52% error rate. You can transform the images, enhance the training set, or incorporate any crazy idea you might have, but the classifier has to be based on k-nn. Using a fancy deep neural net or convolutional net or nonlinear SVM or any other fancy classifier is considered cheating.

Ans: To get $O(n \log n)$ complexity, simply sort the distance to all training points, and pick the smallest k distances (with indices). To get $O(nk)$ complexity, find the smallest distance, record its index and delete it from the training set, repeat k times. To get $O(n)$ complexity, find the k-th smallest distance using the median-of-medians algorithm, then output the distances (and indices) that are smaller than the k-th smallest distance.

One way to speed up the computation for the Euclidean distance is based on the following observation:

$$\|\mathbf{x} - \mathbf{z}\|_2^2 = \|\mathbf{x}\|_2^2 - 2\mathbf{x}^\top \mathbf{z} + \|\mathbf{z}\|_2^2. \tag{1}$$

Thus, to compute the distance between each row of X and each row of Z, all we need is the following three quantities:

$$\mathbf{u} = \mathsf{sum}(X \odot X, 2) \tag{2}$$
$$S = XZ^\top \tag{3}$$
$$\mathbf{v} = \mathsf{sum}(Z \odot Z, 2), \tag{4}$$

based on which we have

$$D^2 = \mathbf{u}\mathbf{1}^\top - 2 * S + \mathbf{1}\mathbf{v}^\top. \tag{5}$$

[In fact, we can omit the last term.]

Exercise 2: Bayes rule and Bayes error (20 pts)

Let (X, Y) be a random sample from some joint distribution, with $X \in \mathcal{X}$ and $Y \in [c] := \{1, \dots, c\}$. Prove that among all rules $f : \mathcal{X} \to [c]$, the Bayes rule (with ties breaking arbitrarily)

$$f^*(X) = \operatorname*{argmax}_{m=1,\dots,c} \mathbf{P}(Y = m|X) \tag{6}$$

achieves the minimum classification error

$$P^* = \mathbf{E}(P^*(X)),\ P^*(X) = 1 - \max_{m=1,\dots,c} \mathbf{P}(Y = m|X). \tag{7}$$

Ans: For any rule $f : \mathcal{X} \to [c]$, we have

$$\begin{aligned}
\mathbf{P}(Y \neq f(X)) &= 1 - \mathbf{P}(Y = f(X)) && (8)\\
&= 1 - \sum_{m=1}^{c} \mathbf{P}(Y = m, f(X) = m) && (9)\\
&= 1 - \sum_{m=1}^{c} \mathbf{E}[\mathbf{P}(Y = m, f(X) = m|X)] && (10)\\
&= 1 - \sum_{m=1}^{c} \mathbf{E}[\mathbf{1}_{f(X)=m}\mathbf{P}(Y = m|X)] && (11)\\
&= 1 - \mathbf{E}\Big[\sum_{m=1}^{c} \mathbf{1}_{f(X)=m}\mathbf{P}(Y = m|X)\Big]. && (12)
\end{aligned}$$

Note that $\sum_{m=1}^{c} \mathbf{1}_{f(X)=m} = 1$. To minimize $\mathbf{P}(Y \neq f(X))$, we need to maximize $\mathbf{E}\Big[\sum_{m=1}^{c} \mathbf{1}_{f(X)=m}\mathbf{P}(Y = m|X)\Big]$, whence follows the claimed formula of the Bayes rule. The formula for the Bayes error is clear.

Exercise 3: ν-Support Vector Machines (50 pts)

Convention: In this exercise, if it helps simplifying your thoughts or presentation, you can always ignore the bias term b.

1. Consider the following "more general" formulation of C-SVM:

$$\min_{\mathbf{w},b}\ \tfrac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^{n}(\rho - y_i\hat{y}_i)_+,\ \text{where } \hat{y}_i = \mathbf{w}^\top\mathbf{x}_i + b, \tag{13}$$

$(t)_+ := \max\{t, 0\}$, and $C \geq 0$ is a tuning hyper-parameter. Prove that (13), with any margin parameter $\rho > 0$, can be reduced to the simpler SVM with $\rho = 1$. [You may change the hyper-parameter C.]

Here by reducing A to B we mean: if we can solve B then we can easily recover a solution for A.

Ans: If we divide the objective function by ρ:

$$\min_{\mathbf{w},b} \ \frac{C}{2\rho}\|\mathbf{w}\|_2^2 + \sum_{i=1}^n (1 - y_i\hat{y}_i/\rho)_+, \ \text{where } \hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b. \tag{14}$$

Next, we perform some change of variable: $C' \leftarrow \rho C$, $\mathbf{w} \leftarrow \mathbf{w}/\rho$, $b \leftarrow b/\rho$:

$$\min_{\mathbf{w},b} \ \frac{C'}{2}\|\mathbf{w}\|_2^2 + \sum_{i=1}^n (1 - y_i\hat{y}_i)_+, \ \text{where } \hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b. \tag{15}$$

Thus, (13) is equivalent as setting $\rho = 1$ but changing C to ρC.

Some people may actually try to change the data $(X, \mathbf{y})$ here, which is less ideal but acceptable.

2. Let us now introduce a penalty for ρ above and try to optimize it:

$$\min_{\mathbf{w},b,\rho} \ \frac{C}{2}\|\mathbf{w}\|_2^2 - \nu\rho + \sum_{i=1}^n (\rho - y_i\hat{y}_i)_+, \ \text{where } \hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b. \tag{16}$$

Use a similar argument as above to prove that we can reduce (16) to the simpler case where $C = 1$. [Note that here ρ is an optimization variable instead of a tuning hyper-parameter.]

Ans: We multiply the objective function by C and then perform the change of variables: $\mathbf{w} \leftarrow C\mathbf{w}$, $\rho \leftarrow C\rho$, $b \leftarrow Cb$.

3. For the following we fix $C = 1$, with $\nu \geq 0$ being the only tuning hyper-parameter:

$$\min_{\mathbf{w},b,\rho} \ \frac{1}{2}\|\mathbf{w}\|_2^2 - \nu\rho + \sum_{i=1}^n (\rho - y_i\hat{y}_i)_+, \ \text{where } \hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b. \tag{17}$$

Derive the Lagrangian dual of (17).

Ans: We re-write the primal problem as:

$$\min_{\mathbf{w},b,\rho,\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|_2^2 - \nu\rho + \sum_{i=1}^n \xi_i \tag{18}$$

$$\text{s.t. } \forall i, \xi_i \geq \rho - y_i(\mathbf{w}^\top \mathbf{x}_i + b) \tag{19}$$

$$\xi_i \geq 0. \tag{20}$$

Next, we introduce Lagrangian multipliers $\boldsymbol{\alpha} \geq \mathbf{0}$ for the first inequality constraint and $\boldsymbol{\beta} \geq \mathbf{0}$ for the second. The Lagrangian is:

$$\frac{1}{2}\|\mathbf{w}\|_2^2 - \nu\rho + \sum_{i=1}^n \xi_i - \alpha_i(\xi_i - \rho + y_i(\mathbf{w}^\top \mathbf{x}_i + b)) - \beta_i\xi_i. \tag{21}$$

Taking derivative wrt b we have

$$\sum_i \alpha_i y_i = 0. \tag{22}$$

Taking derivative wrt ρ we have

$$\sum_i \alpha_i = \nu. \tag{23}$$

Taking derivative wrt ξ_i we have

$$\alpha_i + \beta_i = 1, \tag{24}$$

which is equivalent to $\boldsymbol{\alpha} \leq \mathbf{1}$ since $\boldsymbol{\beta} \geq \mathbf{0}$.

Taking derivative wrt $\mathbf{w}$ we have

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i. \tag{25}$$

Plugging these equalities back to the Lagrangian we have the dual problem:

$$\max_{\mathbf{1} \geq \boldsymbol{\alpha} \geq \mathbf{0}} \; -\frac{1}{2} \Big\| \sum_i \alpha_i y_i \mathbf{x}_i \Big\|_2^2 \tag{26}$$

$$\text{s.t.} \; \sum_i \alpha_i = \nu \tag{27}$$

$$\sum_i \alpha_i y_i = 0. \tag{28}$$

4. To understand what is really going on in (17), consider the following subproblem:

$$\min_{\rho} \; -k\rho + \sum_{i=1}^{n} (\rho - a_i)_+, \tag{29}$$

where $\mathbf{a} \in \mathbb{R}^n$ is given and we have set $\nu = k \in \mathbb{N}$. Solve (29) in closed-form. [Sorting a_i might be helpful. Those familiar with financial risks might recognize CVaR here.]

This is not required, but if you plug your closed-form solution into (17) you will perhaps find an intuitive explanation of the goal of (17).

Ans: W.l.o.g. we assume $a_1 \leq a_2 \leq \cdots \leq a_n$. For any ρ, there exists j such that $a_j \leq \rho \leq a_{j+1}$. Thus,

$$-k\rho + \sum_{i=1}^{n} (\rho - a_i)_+ = -k\rho + \sum_{i=1}^{j} (\rho - a_i) = (j-k)\rho - \sum_{i=1}^{j} a_i. \tag{30}$$

We want to minimize the above objective, which is increasing in ρ if $j \geq k$ and decreasing in ρ if $j \leq k$. Therefore, we can choose $j = k$ so that at the optimum $\rho = a_k$, and the optimal value is $-\sum_{i=1}^{k} a_i$. More generally, for any $\nu \in \mathbb{R}$, let $\lceil \nu \rceil$ be the smallest integer that is larger than ν, then $\rho = a_{\lceil \nu \rceil}$ and the minimum value is

$$-(\nu - \lceil \nu \rceil) a_{\lceil \nu \rceil} - \sum_{i=1}^{\lceil \nu \rceil} a_i. \tag{31}$$

[Another way to derive the optimal value of ρ is to compute the subgradient of the convex function $(\cdot)_+$, so that $0 = -k + \sum_{i=1}^{n} \mathbf{1}_{a_i < \rho} + \sum_{i=1}^{n} t_i \mathbf{1}_{a_i = \rho}$, where each t_i can be any number in the interval $[0, 1]$. Clearly, choose $\rho = a_k$ and t_i appropriately we can satisfy the previous optimality condition. We can also choose $\rho = a_{k+1}$ or any number between a_k and a_{k+1} to satisfy the same optimality condition. Therefore, the optimal ρ is any number in the interval $[a_k, a_{k+1}]$, but the optimal value is always $-\sum_{i=1}^{k} a_k$.]

Plugging the above closed-form solution (31) into (17), we have

$$\min_{\mathbf{w}, b} \; \tfrac{1}{2}\|\mathbf{w}\|_2^2 + (\nu - \lceil \nu \rceil)(-y \odot \hat{y})_{[\lceil \nu \rceil]} + \sum_{i=1}^{\lceil \nu \rceil} (-y \odot \hat{y})_{[i]}, \; \text{where } \hat{y}_i = \mathbf{w}^\top \mathbf{x}_i + b. \tag{32}$$

and we use $a_{[i]}$ to denote the i-th largest element of a. So the algorithm tries to minimize the sum of the ν largest values $-y_i \hat{y}_i$. Note that $-y_i \hat{y}_i$ is large if the prediction $\hat{y}_i$ is very wrong (with large magnitude).

5. Consider the following problem, where we take two sets $P := \{\mathbf{x}_i : y_i = 1\}$ and $N := \{\mathbf{x}_i : y_i = -1\}$ and we want to find the minimal distance between the convex hulls of P and N. Recall that any point $\mathbf{z}$ in

the convex hull of a set $Q := \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$ can be written as $\mathbf{z} = \sum_{i=1}^m \alpha_i \mathbf{z}_i$ with $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\mathbf{1}^\top \boldsymbol{\alpha} = 1$. Thus, we can formulate the minimal (squared) distance between convex hulls of P and Q as:

$$\min_{\boldsymbol{\alpha} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{\alpha} = 1, \boldsymbol{\beta} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{\beta} = 1} \frac{1}{2} \Big\| \sum_{i: y_i = 1} \alpha_i \mathbf{x}_i - \sum_{i: y_i = -1} \beta_i \mathbf{x}_i \Big\|_2^2. \tag{33}$$

Explain how the dual problem in Ex3.3 is related to the minimal distance (33) above. [The two pictures below might help, or not.]

Ans: Upon re-scaling $\boldsymbol{\alpha}$ by $2/\nu$ and splitting $\boldsymbol{\alpha}$ into $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we can rewrite the dual problem in Ex3.3 as:

$$\min_{2/\nu \geq \boldsymbol{\alpha} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{\alpha} = 1, 2/\nu \geq \boldsymbol{\beta} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{\beta} = 1} \frac{1}{2} \Big\| \sum_{i: y_i = 1} \alpha_i \mathbf{x}_i - \sum_{i: y_i = -1} \beta_i \mathbf{x}_i \Big\|_2^2. \tag{34}$$

Comparing this with (33) we see that the only difference is that when we form the convex hulls, we have the additional constraint $2/\nu \geq \boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0$. [It is clear that we should choose $\frac{2}{\nu}(n_- \wedge n_+) \geq 1$, i.e., $\nu \leq 2(n_- \wedge n_+)$.] When $\nu \in (0, 2]$, the two problems are actually identical, while for $\nu \in (2, 2(n_- \wedge n_+)]$, the dual of Ex3.3 finds the closest pair of points in the "reduced convex hull," i.e. each convex combination coefficient is upper bounded by $2/\nu$.

[To understand the reduced convex hull, take two points or three points and construct the convex hull and the reduced convex hull by hand. An example can be found in "A geometric interpretation of ν-SVM classifiers, Crisp and Burges, NIPS'99, Figure 1."]