COMS 4771 Machine Learning (Spring 2020)
Problem Set #5

Joseph High - `jph2185@columbia.edu`

May 6, 2020

Problem 1: Improving the Confidence

Proof. Using the hint, one can construct an algorithm (referred to as algorithm $\mathcal{B}$, or just $\mathcal{B}$, henceforth) that calls algorithm $\mathcal{A}$ on k independent sets of training samples where each run will be done over a draw of $n = O\left(\frac{1}{\epsilon^2}\right)$ samples. In total, $\mathcal{B}$ is run over a draw of n' samples such that the error of the resulting model is within ϵ-distance of the model with minimum error over all $f \in \mathcal{F}$. Algorithm $\mathcal{B}$ is constructed below.

Fix an $\epsilon_1 = \dfrac{\epsilon}{2}$ such that with probability 0.55 we have the following for each run of $\mathcal{A}$ (over a draw of n samples in each run):

$$\operatorname{err}(f_n^{\mathcal{A}}) - \inf_{f \in \mathcal{F}} \operatorname{err}(f) \ \leq\ \epsilon_1 \quad \Longleftrightarrow \quad \operatorname{err}(f_n^{\mathcal{A}}) \ \leq\ \inf_{f \in \mathcal{F}} \operatorname{err}(f) + \epsilon_1 \tag{1}$$

Fix a $\delta > 0$ and choose some $\gamma_1 \leq \frac{\delta}{2}$ such that with probability $1 - \gamma_1 \ \geq\ 1 - \frac{\delta}{2}$ (i.e., with probability at least $1 - \frac{\delta}{2}\big)$, the k iterations of algorithm $\mathcal{A}$ will return at least one model with a generalization error within $\frac{\epsilon}{2}$ of $\inf\limits_{f \in \mathcal{F}} \operatorname{err}(f)$. Henceforth, such a model will be referred to as a "good" model, while models that don't exhibit this behavior will be referred to as "bad". Let $\mathcal{F}_{\mathcal{B}}$ be the collection of the k models returned by algorithm $\mathcal{A}$. That is, for $i \in \{1, \ldots, k\}$, let $f_{n_i}^{\mathcal{A}}$ denote the model returned by $\mathcal{A}$ in the i^{th} run, then $\mathcal{F}_{\mathcal{B}} = \left\{f_{n_1}^{\mathcal{A}}, \ldots, f_{n_k}^{\mathcal{A}}\right\}$. The number of models in $\mathcal{F}_{\mathcal{B}}$ (i.e., the number of iterations of $\mathcal{A}$) should be sufficiently large so that the probability of at least one "good" model being returned is at least $1 - \frac{\delta}{2}$. To determine a sufficient k, consider the following:

Let E_i be the random variable that denotes the event that the i^{th} model, $f_{n_i}^{\mathcal{A}}$, is a "good" model. That is, let $E_i = \left\{\operatorname{err}\left(f_{n_i}^{\mathcal{A}}\right) - \inf_{f \in \mathcal{F}} \operatorname{err}(f) \ \leq\ \frac{\epsilon}{2}\right\}$. Then, the complement, E_i^{c}, is the event that $f_{n_i}^{\mathcal{A}}$ is a "bad" model: $E_i^{\mathrm{c}} = \left\{\operatorname{err}\left(f_{n_i}^{\mathcal{A}}\right) - \inf_{f \in \mathcal{F}} \operatorname{err}(f) \ >\ \frac{\epsilon}{2}\right\}$. From the supposition, the probability that the i^{th} run of $\mathcal{A}$ returns a "bad" model is

$$\mathbb{P}\left(E_i^{\mathrm{c}}\right) = 1 - \mathbb{P}\left(E_i\right) = 1 - 0.55 = 0.45$$

Then, the probability that all k models returned by $\mathcal{A}$ are "bad" is

$$\mathbb{P}\left(\forall i \in \{1,\ldots,k\} : \; f_{n_i}^{\mathcal{A}} \in \mathcal{F}_{\mathcal{B}} \text{ is "bad"}\right) \;=\; \mathbb{P}\left(\bigcap_{i=i}^{k} E_i^{\mathrm{c}}\right) \;=\; \prod_{i=1}^{k} \mathbb{P}\left(E_i^{\mathrm{c}}\right) = \prod_{i=1}^{k} 0.45 = (0.45)^k$$

Then, the probability that there is *at least one* "good" model in $\mathcal{F}_{\mathcal{B}}$ is

$$\mathbb{P}\left(\exists i \in \{1,\ldots,k\} : \; f_{n_i}^{\mathcal{A}} \in \mathcal{F}_{\mathcal{B}} \text{ is "good"}\right) \;=\; 1 - \mathbb{P}\left(\bigcap_{i=i}^{k} E_i^{\mathrm{c}}\right) \;=\; 1 - (0.45)^k$$

Therefore, we need to choose a k such that

$$1 - (0.45)^k \;\geq\; 1 - \frac{\delta}{2} \quad\iff\quad (0.45)^k \;\leq\; \frac{\delta}{2} \quad\iff\quad k\ln(0.45) \;\leq\; \ln\left(\frac{\delta}{2}\right)$$

$$\iff\quad -k\ln\left(\frac{1}{0.45}\right) \;\leq\; -\ln\left(\frac{2}{\delta}\right) \quad\iff\quad k \;\geq\; \frac{\ln\left(\frac{2}{\delta}\right)}{\ln\left(\frac{1}{0.45}\right)}$$

It suffices to set $k = \left\lceil \frac{\ln(2/\delta)}{\ln(1/0.45)} \right\rceil$. Indeed, k is necessarily an integer, so the ceiling function should be applied to the right-hand side.

Since $\mathcal{F}_{\mathcal{B}}$ is finite, the ERM algorithm can be applied to $\mathcal{F}_{\mathcal{B}}$. Then, for a fixed confidence level $\gamma_2 \leq \frac{\delta}{2}$ and tolerance $\epsilon_2 = \frac{\epsilon}{2}$, apply the ERM algorithm over $\mathcal{F}_{\mathcal{B}}$ using a test set of size m. By Occam's Razor, such an m should be such that[1]

$$m \;\geq\; \frac{1}{2\epsilon_2^2}\ln\left(\frac{2|\mathcal{F}_{\mathcal{B}}|}{\gamma_2}\right) \;\geq\; \frac{1}{2(\epsilon/2)^2}\ln\left(\frac{2k}{\delta/2}\right) \;=\; \frac{2}{\epsilon^2}\ln\left(\frac{4k}{\delta}\right)$$

and since m is necessarily a positive integer, we can set $m = \left\lceil \frac{2}{\epsilon^2}\ln\left(4k/\delta\right)\right\rceil$.
Let f_m^{ERM} be the model returned by the ERM algorithm. If follows from Occam's Razor that, with probability at least $1 - \frac{\delta}{2}$,

$$\operatorname{err}\left(f_m^{ERM}\right) - \inf_{f \in \mathcal{F}_{\mathcal{B}}} \operatorname{err}(f) \;\leq\; \epsilon_2 \tag{2}$$

Now, the probability that algorithm $\mathcal{B}$ fails is equal to the probability that either algorithm $\mathcal{A}$ fails to return a "good" model in k iterations or the ERM algorithm fails to select a "good" model from $\mathcal{F}_{\mathcal{B}}$. That is,

$$\begin{aligned}
\mathbb{P}(\mathcal{B}\text{ fails}) \;&=\; \mathbb{P}\left(\left\{\forall f_{n_i}^{\mathcal{A}} \in \mathcal{F}_{\mathcal{B}} \;:\; \operatorname{err}\left(f_{n_i}^{\mathcal{A}}\right) - \inf_{f\in\mathcal{F}}\operatorname{err}(f) > \epsilon_1\right\} \bigcup \left\{\operatorname{err}\left(f_m^{ERM}\right) - \inf_{f\in\mathcal{F}_{\mathcal{B}}}\operatorname{err}(f) > \epsilon_2\right\}\right) \\
&\leq\; \mathbb{P}\left\{\forall f_{n_i}^{\mathcal{A}} \in \mathcal{F}_{\mathcal{B}} \;:\; \operatorname{err}\left(f_{n_i}^{\mathcal{A}}\right) - \inf_{f\in\mathcal{F}}\operatorname{err}(f) > \epsilon_1\right\} \;+\; \mathbb{P}\left\{\operatorname{err}\left(f_m^{ERM}\right) - \inf_{f\in\mathcal{F}_{\mathcal{B}}}\operatorname{err}(f) > \epsilon_2\right\} \\
&=\; \gamma_1 \;+\; \gamma_2 \;\leq\; \frac{\delta}{2} \;+\; \frac{\delta}{2} \;=\; \delta
\end{aligned}$$

[1] The specific inequality used here was extracted from the proof of efficient PAC learnability for finite $\mathcal{F}$, using the the ERM algorithm, provided in the Learning Theory lecture notes (slide 8).

where the second inquality is a result of Boole's inequality (a.k.a. subadditivity).
Then, by combining inequalities (1) and (2), we have, with probability at least $1-\delta$,

$$\begin{aligned}
\operatorname{err}(f_m^{ERM}) &\le \epsilon_2 + \inf_{f\in\mathcal{F}_\mathcal{B}} \operatorname{err}(f) \\
&\le \epsilon_2 + (\epsilon_1 + \inf_{f\in\mathcal{F}} \operatorname{err}(f)) \\
&= \frac{\epsilon}{2} + \frac{\epsilon}{2} + \inf_{f\in\mathcal{F}} \operatorname{err}(f)) \\
\implies \quad \operatorname{err}(f_m^{ERM}) - \inf_{f\in\mathcal{F}} \operatorname{err}(f)) &\le \epsilon
\end{aligned}$$

Since f_m^{ERM} is an arbitrary model returned by algorithm $\mathcal{B}$, the above inequality holds for any model returned by algorithm $\mathcal{B}$, with probability at least $1-\delta$. Note, each run of algorithm $\mathcal{B}$ is over a draw of $n' = kn + m$ samples. Indeed, $k \cdot n$ samples are used for the k iterations of $\mathcal{A}$ and m test samples are used to identify the model in $\mathcal{F}_\mathcal{B}$ with the minimum empirical error. Then, denote any model returned by $\mathcal{B}$ as $f_{n'}^{\mathcal{B}}$. Hence, for a tolerance $\epsilon > 0$, with probability at least $1-\delta$, we have

$$\operatorname{err}(f_{n'}^{\mathcal{B}}) - \inf_{f\in\mathcal{F}} \operatorname{err}(f)) \le \epsilon$$

Moreover,

$$k = \left\lceil \frac{\ln(2/\delta)}{\ln(1/0.45)} \right\rceil \implies k = O(\ln(1/\delta))$$

$$m = \left\lceil \frac{2}{\epsilon^2} \ln(4k/\delta) \right\rceil \implies m = O\left(\frac{1}{\epsilon^2}\ln(1/\delta)\right)$$

Recall that, $n = O\left(\frac{1}{\epsilon^2}\right)$. Therefore, since $n' = nk + m \implies n' = O\left(\frac{1}{\epsilon^2}\ln(1/\delta)\right)$.
Since $\ln(1/\delta)$ grows at a slower rate than $1/\delta$ (i.e., $\ln(1/\delta) \le 1/\delta$), then n' is bounded by a polynomial in both $1/\epsilon$ and $1/\delta$. That is, $n' = O\left(\frac{1}{\epsilon^2}\ln(1/\delta)\right) \in \operatorname{poly}\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$, or equivalently

$$n' = \operatorname{poly}\left(\frac{1}{\epsilon}, \frac{1}{\delta}\right)$$

Hence, $\mathcal{F}$ is *efficiently* PAC-learnable.

□

Problem 2: Non-linear Dimensionality Reduction

(i) For some $i \in \{1, \ldots, n\}$, the derivative of the objective function with respect to y_i is as follows:

$$\begin{aligned}
\frac{\partial}{\partial y_i} \sum_{i,j} (\|y_i - y_j\| - \pi_{ij})^2 &= \sum_{i,j} \frac{\partial}{\partial y_i} \left[(\|y_i - y_j\| - \pi_{ij})^2 \right] \\
&= \sum_{\substack{i,j \\ j \neq i}} 2 (\|y_i - y_j\| - \pi_{ij}) \cdot \frac{\partial}{\partial y_i} \Big[\|y_i - y_j\| - \pi_{ij} \Big] \\
&= 2 \sum_{\substack{i,j \\ j \neq i}} (\|y_i - y_j\| - \pi_{ij}) \cdot \frac{y_i - y_j}{\|y_i - y_j\|} \\
&= 2 \sum_{\substack{i,j \\ j \neq i}} \left(1 - \frac{\pi_{ij}}{\|y_i - y_j\|} \right) (y_i - y_j)
\end{aligned}$$

(ii) The optimization problem is *non-convex*. While the feasible region, $\mathbb{R}^d$, is a convex set (since the problem is unconstrained), the objective function is not a convex function. To see this, consider the following:

$$\sum_{i,j} (\|y_i - y_j\| - \pi_{ij})^2 = \sum_{i,j} \|y_i - y_j\|^2 - 2\pi_{ij} \|y_i - y_j\| + \pi_{ij}^2$$

Without loss of generality, consider one summand

$$\underbrace{\|y_i - y_j\|^2}_{\text{convex}} - \underbrace{2\pi_{ij} \|y_i - y_j\| + \pi_{ij}^2}_{\text{convex}}$$

Note that all p-norms are convex functions, provided that $p \geq 1$. Now, because the π_{ij} terms denote the shortest path between data points x_i and x_j, they're deterministic scalars, and so each summand is the difference of convex functions, which is the equivalent to the sum of a convex function and a concave function. The difference of convex functions is not necessarily convex, but it could be.

Counter-example: Let $f(y_1) = \sum_{i,j} (\|y_i - y_j\| - \pi_{ij})^2$. Without loss of generality, let $d = 1$, that is, $y_j \in \mathbb{R}^1$ and suppose that $n = 3$. Fix $y_2 = 0$ and $y_3 = 1$, and suppose $\pi_{12} = \pi_{13} = 1$. The objective is then

$$f(y_1) = \sum_{j=2}^{3} (\|y_i - y_j\| - \pi_{1j})^2 = (\|y_1\| - 1)^2 + (\|y_1 - 1\| - 1)^2$$

Recall that a function g is convex if and only if for all $x, y \in \mathbf{dom}(g)$,

$$g(y) \geq g(x) + g'(x)(y - x)$$

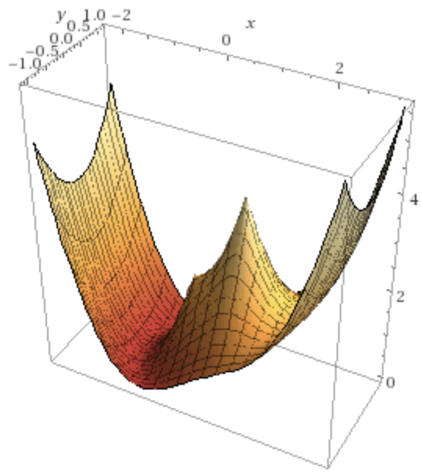
Proceeding with this in mind, consider f at $y_1 = \frac{1}{2}$ and $y_1 = \frac{5}{4}$:

$$f(1/2) = (\|1/2\| - 1)^2 \; + \; (\|1/2 - 1\| - 1)^2 = 1/2$$
$$f(5/4) = (\|5/4\| - 1)^2 \; + \; (\|5/4 - 1\| - 1)^2 = 5/8$$

The derivative of the objective was computed in part (i), which will be used here to compute $f'(2)$:

$$f'(1/2) = 2 \cdot \left[\left(1 - \frac{1}{\|1/2\|}\right)(1/2) + \left(1 - \frac{1}{\|1/2 - 1\|}\right)(1/2 - 1) \right] = 2 \cdot (1/4 - 3/4) = 0$$

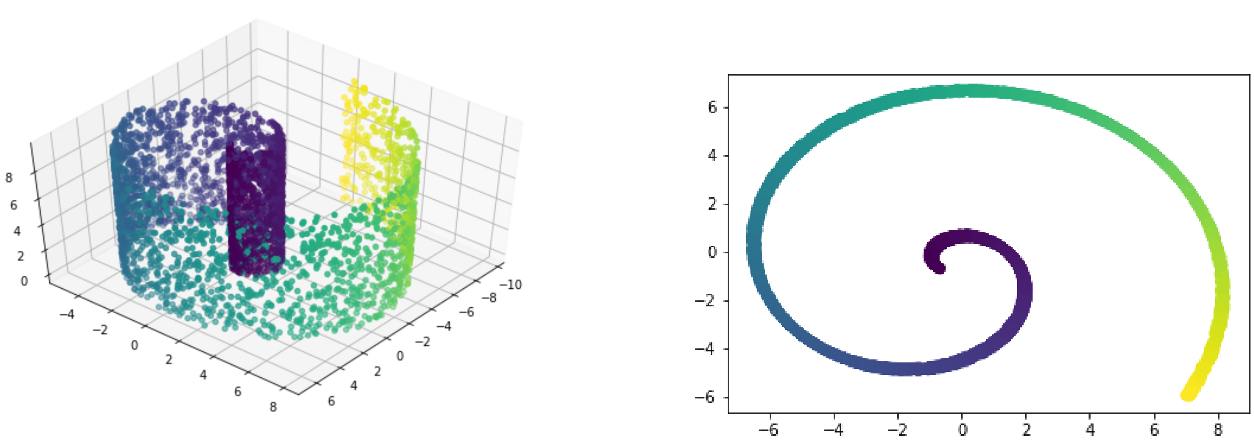
Then, $f(2) + f'(2) \cdot (3 - 2) = 1 + 2 = 3$



(iii) Code submitted to Courseworks.

(iv) Below are the results of PCA on the `swiss_roll.txt` data set (implemented in Python).

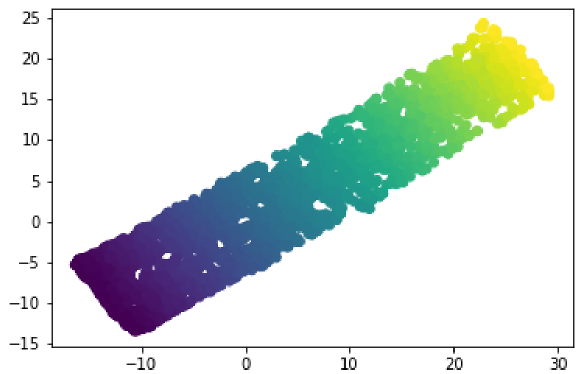
Swiss Roll Results (`swiss_roll.txt`)



(a) 3-Dimensional Swiss Roll (b) PCA Embedding

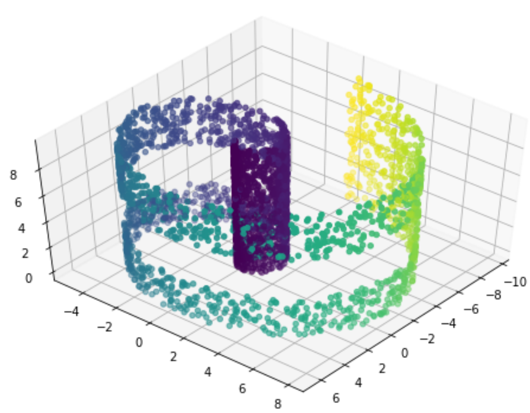
Below are the results of my implementation of the non-linear embedding algorithm given in the homework on the `swiss_roll.txt` data set. The non-linear embedding succeeded and appears to have done far better than PCA.

Figure 1: Low-Dimensional Embedding Result (`swiss_roll.txt`)

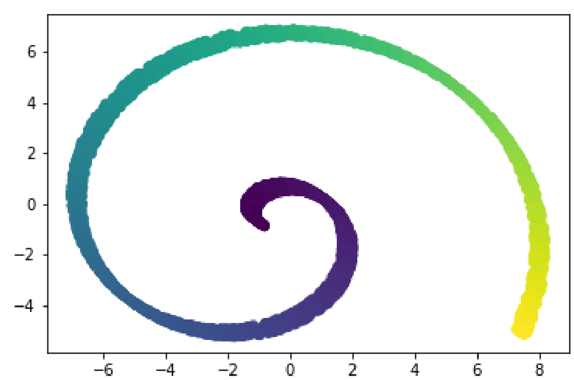


Swiss Roll Hole Results (`swiss_roll_hole.txt`)

Below are the results of PCA on the `swiss_roll_hole.txt` data set (implemented in Python).



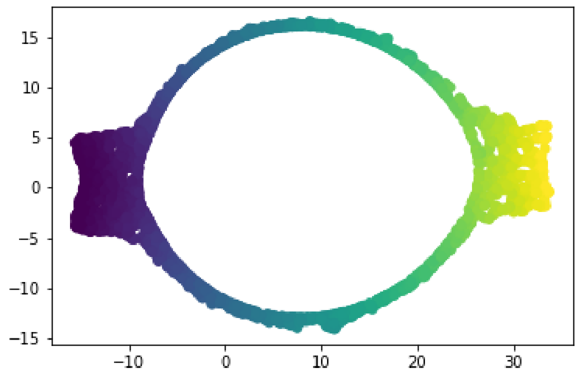
(a) 3-Dimensional Swiss Roll with hole



(b) PCA Embedding of Swiss Roll with hole

Below are the results of my implementation of the non-linear embedding algorithm given in the homework on the `swiss_roll_hole.txt` data set. Similar to the result for the other data set, the non-linear embedding succeeded and appears to have done far better than PCA.

Figure 2: Low-Dimensional Embedding Result (`swiss_roll_hole.txt`)



When the learning rate was set at 0.01, the non-linear embedding failed to capture sufficient information from the swiss roll. However, when the learning rate was adjusted to 0.0001, the nonlinear embedding improved, significantly.