# 10-701 Introduction to Machine Learning

## Homework 4, version 1.0

*Due Nov 13, 11:59 am*

---

### Rules:

1. Homework submission is done via CMU Autolab system. Please package your writeup and code into a zip or tar file, *e.g.*, let `submit.zip` contain `writeup.pdf` and the code. Submit the package to https://autolab.cs.cmu.edu/courses/10701-f15.
2. Like conference websites, repeated submission is allowed. So please feel free to refine your answers. We will only grade the latest version.
3. Autolab may allow submission after the deadline, note however it is because of the late day policy. Please see course website for policy on late submission.
4. We recommend that you typeset your homework using appropriate software such as LaTeX. If you are writing please make sure your homework is cleanly written up and legible. The TAs will not invest undue effort to decrypt bad handwriting.
5. You are allowed to collaborate on the homework, but you should write up your own solution and code. Please indicate your collaborators in your submission.

---

# 1 VC dimension (20 Points) (Xun)

To show a concept class $H$ has VC dimension $d$, we need to prove both the lower bound $\text{VCdim}(H) \geq d$ and the upper bound $\text{VCdim}(H) \leq d$.

1. Show that linear classifiers $h(x) = \mathbf{1}_{\{a^\top x + b \geq 0\}}$ in $\mathbb{R}^n$ has VC dimension $n+1$.

   Hint: the following theorem might be useful in proving the upper bound. A set of $n+2$ points in $\mathbb{R}^n$ can be partitioned into two disjoint subsets $S_1$ and $S_2$ such that their convex hulls intersect. The convex hull $\mathbf{conv}(C)$ of a set $C$ is defined as the set of all convex combinations of points in $C$:

   $$\mathbf{conv}(C) = \left\{ \sum_{i=1}^{k} \alpha_i x_i : x_i \in C, \alpha_i \geq 0, \sum_{i=1}^{k} \alpha_i = 1 \right\}. \tag{1}$$

   You do not need to know anything about convexity beyond this hint to solve this problem.

2. Show that axis-aligned boxes $h(x) = \mathbf{1}_{\{a_i \leq x_i \leq b_i, \forall i\}}$ in $\mathbb{R}^n$ has VC dimension $2n$.

# 2 AdaBoost (30 Points) (Xun)

Consider $m$ training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x \in \mathcal{X}$ and $y \in \{-1, 1\}$. Suppose we have a weak learning algorithm $A$ which produces a hypothesis $h : \mathcal{X} \to \{-1, 1\}$ given any distribution $D$ of examples. AdaBoost works as follows (slightly different from the lecture slides, but they are equivalent):

- Begin with a uniform distribution $D_1(i) = \frac{1}{m}$, $i = 1, \ldots, m$.
- At each round $t = 1, \ldots, T$,
  - Run $A$ on $D_t$ and get $h_t$.
  - Update $D_{t+1}(i) = \frac{D_t}{Z_t} e^{-\alpha_t y_i h_t(x_i)}$, where $Z_t$ is the normalizer and $i = 1, \ldots, m$.

Note that since $A$ is a weak learning algorithm, the produced $h_t$ at round $t$ is only slightly better than random guessing, say, by a margin $\gamma_t$:

$$\epsilon_t = \mathsf{err}_{D_t}(h_t) = \Pr_{x \sim D_t}[y \neq h_t(x)] = \frac{1}{2} - \gamma_t. \tag{2}$$

In the end, AdaBoost outputs $H = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t\right)$ as the learned hypothesis. We will now prove that the training error $\mathsf{err}_S(H)$ of AdaBoost decreases to zero at a very fast rate. In the answer, please state clearly why the derivation makes sense, for instance "by Cauchy-Schwarz, ...".

1. Let's first justify the update rule. Imagine there is an adversarial who wants to fool $h_t$ in the next round by adjusting the distribution. More formally, given $h_t$, the adversarial wants to set $D_{t+1}$ such that $\mathsf{err}_{D_{t+1}}(h_t) = \frac{1}{2}$. Show that the particular choice of $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$ achieves this goal.

   Note: why do we want such an adversarial setting? Because otherwise $A$ might as well return $h_t$ or $-h_t$ again in round $t+1$ and still be slightly better than random guessing, which means it essentially learns nothing.

2. Show that $D_{T+1}(i) = \left(m \cdot \prod_{t=1}^{T} Z_t\right)^{-1} e^{-y_i f(x_i)}$, where $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

3. Show that $\mathsf{err}_S(H) \leq \prod_{t=1}^{T} Z_t$.

4. Show that $\prod_{t=1}^{T} Z_t \leq e^{-2\sum_{t=1}^{T} \gamma_t^2}$.

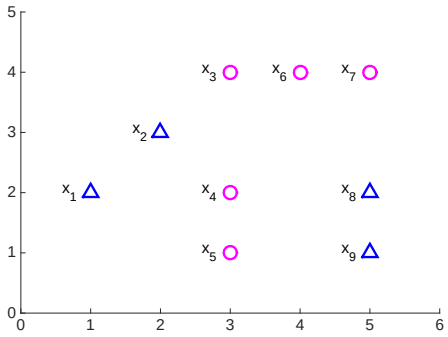


Figure 1: Toy data for AdaBoost.

5. Now let $\gamma = \min_t \gamma_t$. From 3 and 4, we know the training error approaches zero at exponential rate with respect to $T$. Then how many rounds are needed to achieve a training error $\varepsilon > 0$? Please express in big-O notation, $T = \mathcal{O}(\cdot)$.

6. Consider the data set in Figure 1. Run $T = 3$ iterations of AdaBoost with decision stumps (axis-aligned separators) as the base learners. Illustrate the learned weak hypotheses $\{h_t\}$ in Figure 1 and fill in Table 1. The MATLAB code that generates Figure 1 is available on the course website.

   We recommend writing a simple program as it might be tedious to calculate by hand. It will also help you understand how it works in practice.

| $t$ | $\epsilon_t$ | $\alpha_t$ | $D_t(1)$ | $D_t(2)$ | $D_t(3)$ | $D_t(4)$ | $D_t(5)$ | $D_t(6)$ | $D_t(7)$ | $D_t(8)$ | $D_t(9)$ | $\mathsf{err}_S(H)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | |

Table 1: AdaBoost results

# 3 Gaussian Mixture Model (10 Points) (Hao)

Consider a multivariate Gaussian Mixture Model with $K$ components:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \tag{3}$$

1. Show that $\mathbb{E}[x] = \sum_k \pi_k \mu_k$.
2. Show that $\mathsf{Cov}[x] = \sum_k \pi_k [\Sigma_k + \mu_k \mu_k^\top] - \mathbb{E}[x]\mathbb{E}[x]^\top$.

# 4 K-means (40 Points) (Hao)

Given $n$ data samples in $\mathcal{X} \subseteq \mathbb{R}^d$ and an integer $K$, we showed in class that the K-means algorithm tries to determine $K$ clusters $\{C_k\}_{k=1}^K$ with centers $\mathcal{U}_K = \{\mu_k\}_{k=1}^K \subseteq \mathbb{R}^d$, and a mapping function $f : \mathcal{X} \to \{1, \cdots, K\}$

which assigns each $x_i \in \mathcal{X}$ to one of the clusters, so as to optimize the following objective,

$$\phi = \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} \|x_{ki} - x_{kj}\|^2 \tag{4}$$

where $x_{ki}$ denotes the $i$th sample in $C_k$ and $n_k$ is the number of data samples in $C_k$.

## 4.1 Theory

1. Prove the following Lemma.

   **Lemma 1.** *Given a set of points $\mathcal{X} \subseteq \mathbb{R}^d$ with their center as $\bar{x}$. For any point $s$,*

   $$\sum_{x \in \mathcal{X}} \|x - s\|^2 - \sum_{x \in \mathcal{X}} \|x - \bar{x}\|^2 = |\mathcal{X}| \cdot \|\bar{x} - s\|^2 \tag{5}$$

2. Use Lemma.1 to prove that minimizing the objective in Eq.4 is equal to minimizing the following objective:

   $$\omega(\mathcal{U}_K, f; \mathcal{X}) = \sum_{k=1}^{K} \sum_{i=1}^{n} \mathbb{1}(f(x_i) = k) \|x_i - \mu_k\|^2 \tag{6}$$

3. Algorithm.1 presents how K-means proceeds. Show respectively that both **Step 1** and **Step 2** will decrease the objective $\phi$ (or $\omega$).

**Algorithm 1:** K-means Algorithm

```
Initialize {μ_k}_{k=1}^K (randomly, if necessary).
repeat
    Step 1: Decide the class memberships of {x_i}_{i=1}^n by assigning each of them to its nearest cluster
    center.
    Step 2: For each k ∈ {1, ··· , K}, set μ_k to be the center of mass of all points in C_i:
    μ_i = (1/n_k) Σ_{i=1}^{n_k} x_ki
until the objective no longer changes;
```

4. Let $\Omega(K) = \min_{\mathcal{U}_K, f} \omega(\mathcal{U}_K, f; \mathcal{X})$. Show that $\Omega$ is non-increasing in $K$.

5. In K-means (as in Algorithm.1), we terminate the iterative process when the objective no longer changes. Prove that K-means terminates in a finite number of iterations.

## 4.2 Implementation

Now you are ready to implement K-means by yourself. A dataset including 2429 human faces is provded in the file *kmeans_data.csv*. Each of the 2429 lines in this file corresponds to a $19 \times 19$ image of a human face. Every image is represented as a 361-dimensional vector of grayscale values, in column-major format.

1. Implement K-means algorithm, as detailed in Alogorithm.1. Your implementation should initialize $\{\mu_k\}_{k=1}^K$ by uniformly randomly choosing from $\mathcal{X}$. Compute the objective value in Eq.4 of each iteration. You K-means algorithm should be terminated when a given number of iterations $M$ are reached.

2. Run your implementation for 15 times, using $k = 5, M = 50$. Draw the objective v.s. iterations for all 15 runs in one plot. Have they converged? How many iterations does each iteration take to converge? Choose the run with minimal objective value, compute the mean faces for this run, i.e., the centers of the clusters. Visualize the mean faces.

3. Usually the clustering results by K-means can be greatly improved by carefully choosing an initialization strategy. K-means++ is a randomized seeding technique which can improve both the speed and the accuracy of K-means [1]. Algorithm.2 elaborates how K-means++ initializes the clustering centers $\{\mu_k\}_{k=1}^K$.

**Algorithm 2:** K-means++ Initialization

1 Take one center $\mu_1$, chosen uniformly at random from $\mathcal{X}$.

2 Take a new center $\mu_k (k > 1)$ from $\mathcal{X}$, so that $Pr(\mu_k = x_i) = \frac{D(x_i)^2}{\sum_{j=1}^n D(x_j)^2}$ where $D(x)$ is the distance from $x$ to its nearest center in $\{\mu_k\}_{k=1}^{j-1}$.

3 Repeat the above step until all $\{\mu_k\}_{k=1}^K$ have been initialized.

Implement K-means++ based on your K-means implementation. Note that you need to implement a sampler by yourself which samples from a multinomial distribution. Then, run your implementation with K-means++ for 15 times, using $k = 5, M = 50$. Draw the objective v.s. iterations for all 15 runs in one plot. How many iterations do they take to converge? Compute the mean faces for the run with the minimal objective. Visualize the mean faces. Compare your curve and mean faces to your previous ones. Conclude your observation.

Submit both the write-up and your code.

# References

[1] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 5