# Homework 1
# Computer Science
# Fall 2013
# B565

Professor Dalkılıç
Due Sept 4, 2013 11:00 p.m.

August 28, 2013

## Directions

Please follow the syllabus guidelines in turning in your homework.

## Problems

1. Define the following terms
   (a) data mining
   (b) machine learning
   (c) probability
   (d) statistics
   (e) pattern
   (f) consistency
   (g) prediction
   (h) feature
   (i) random variable

2. This problem will illustrate the description of machine learning (ML) given in lecture. Recall that there exists data in a set of pairs

$$\Delta \quad = \quad \{(\delta_1, \ell_1), (\delta_2, \ell_2), \ldots, (\delta_n, \ell_n)\} \tag{1}$$

where the domain $\delta$ is a tuple of values (called features) and the co-domain $\ell$ is usually single value (called the label). Here are a couple examples:

$$\Delta_1 \quad = \quad \{((1,2), \texttt{Y}), ((3,0), \texttt{N})\} \tag{2}$$
$$\Delta_2 \quad = \quad \{(424122, (\texttt{Y}, \blacksquare)), (536235, (\texttt{A}, \square)), (23, \texttt{B}, \circ))\} \tag{3}$$
$$\tag{4}$$

We assume that there exists some function $f : \delta \rightarrow \ell$. For $\Delta_1$, this would mean $f$ maps $(1, 2)$ to $\texttt{Y}$, or more succinctly, $f((1, 2)) = \texttt{Y}$. For $\delta_2$, $f(536235) = (\texttt{A}, \square)$. In ML, analytically determining $f$ is not possible. Instead, we iteratively produce a sequence of approximations to $f$, *i.e.,* $\hat{f}_0, \hat{f}_1, \hat{f}_2, \ldots, \hat{f}_k$ that:

- converge
- are *tolerable*

In other words, $\hat{f}_1$ is at least as good as $\hat{f}_0$ in mapping values, $\hat{f}_2$ is at least as good as $\hat{f}_1$ and so forth. Convergence means we're steadily improving albeit sometimes slowly. "Tolerating" (not a standard ML term) a function means that we accept $f$ will be wrong on some values.

A bare bones ML algorithm is then:

**Input** $\Delta$
**Output** $\hat{f}$
1. Let $\mu$ be measure of correctness of $\hat{f}$
2. Let $\tau$ be some acceptable threshold of error
3. Let $T$ be the number of iterations you're willing to run
4. Let $\mathsf{Build}$ be some function that builds $\hat{f}$
5. $i \leftarrow 0$
6. Initialize $\hat{f}_i$
**while** $(\mu(\hat{f}_i) \geq \tau \ \wedge i < T)$ **do**
  7a. $\hat{f}_{i+1} \leftarrow \mathsf{Build}(\hat{f}_i, \Delta)$
  7b. $i \leftarrow i + 1$
**end while**
**return** $\hat{f}_i$

Some comments are in order.

1. You need to decide when the function is correct. The most obvious answer is observing whether $f(\delta) = \hat{f}(\delta)$; however, this is usually too naïve. Consider matching two names, `thomas Smith` to `Tom ssmith`. Clearly these are the same name, but they aren't identical. Another example is to find the value of $f(0)$ for some continuous function $f$ over $[a, b]$ where $f(a) > 0 > f(b)$. How close is close enough?

2. Once you've decided on what counts as correct, you then need to decide *overall* what's tolerable. Don't be greedy, but understand the consequences of a function that's not always right.

3. So, convergence is seldom guaranteed and even harder to formally prove. This trick is akin to holding your breath.

4. More parameters can and are sent to $\mathsf{Build}$.

6. Be wary of these "starting points." Most of the time, it involves what we computer scientists call "choice" or "pick." In other words, this is gambling.

To make computational sense of this, here's the final problem.

Assume you have a Cartesian grid of only non-negative numbers (quadrant I) of size $[0, 99] \times [0, 99]$. Your going to generate data and labels, find $\hat{f}$, and then witness $\hat{f}$ in action. For this problem, assume all probabilities are uniform.

**Input** $\mu$, $\tau$, $T$
**Output** `REPORT`
1. Generate a random natural number $k \in [5, 15]$.
2a. Generate $k$ unique random triples $((x, y), z)$ where $x, y \in [0, 99]$ and $z \in \{0, 1\}$.
2b. This is $\Delta = \{(x_0, y_0), z_0), \ldots, ((x_{k-1}, y_{k-1}), z_{k-1}))\}$
3a. Pick some point $p \in [0, 99] \times [0, 99]$.
3b. Of the eight slopes $m$ of $0°$, $45°$, $90°$, $\ldots$, $315°$
3c. initialize $\hat{f}_0$ with $p, m$ that best classifies $\Delta$
**while** $(\mu(\hat{f}_i) \geq \tau \ \wedge i < T)$ **do**
  7a. $\hat{f}_{i+1} \leftarrow \mathsf{Build}(\hat{f}_i, \Delta)$
  7b. $i \leftarrow i + 1$
**end while**
**return** `REPORT`

Your report should be a LaTeX document that has two tables: a $\Delta$ table and an $\hat{f}$ table.

| $\Delta$ | |
|---|---|
| $\delta$ | $\ell$ |
| $x_0, y_0$ | $z_0$ |
| $\vdots$ | $\vdots$ |

| $\hat{f}$ | Correctness |
|---|---|
| $\hat{f}_0$ | $\mu(\hat{f}_0)$ |
| $\vdots$ | $\vdots$ |
| $\hat{f}_j$ | $\mu(\hat{f}_j)$ |

Of course, $0 \leq j < T$. Name this program `MLfall2013`.

(a) Assume the data are linearly separable; that is, a hyperplane exists that has 0s on one side and 1s on the other. Give a proof that this random ML algorithm will eventually converge to a correct classifier.

(b) Discuss how you are designing and implementing $\mu$ and $\tau$.

(c) Assume that the label has three distinct values, instead of two. How could you reasonably *easily* modify this algorithm to distinguish three classes?

(d) Discuss potential problems with the classification of the hyperplane.

(e) The method of generating data is artificial – maybe too much so. Explain

(f) Imagine increasing $\delta$ many fold; perhaps $\delta = \mathbb{N}^{100000}$. A curious phenomenon occurs–as the dimensions increase there's less apparent difference between the concepts of near and far.