Introduction to Machine Learning (67577)

Problem Set 2 – PAC Learning and VC Dimension

Due: 27.4.2017

Submission guidelines

- The assignment is to be submitted via Moodle only.
- Files should be zipped to ex_N_First_Last.zip (In this case N stands for 1, First is your first name, Last is your last name. In English, of course).
- The .zip file size should not exceed 10Mb.
- All answers should be submitted as a single pdf file. We encourage you to typeset your answers (either LaTeX, Lyx or Word), but it is not mandatory. Note that if you choose to scan a handwritten solution, then answers written in incomprehensible handwriting or scanned in poor quality will be counted as wrong answers. Moreover, the scans should be integrated with the plots to a single coherent pdf file.
- All of your Python code files used in the practical question should be attached in the zip file. There should be one .py file, other then that there are no special requirements or guidelines for your code. Note that if your code doesn't run on the CS environment you will not get points for this question.

1 Bayes Optimal Classifier

Consider a domain $\mathcal{X}$, the label set $\mathcal{Y} = \{0, 1\}$, and the zero-one loss:

$$l^{0-1}(h, (x, y)) = \begin{cases} 0 & h(x) = y \\ 1 & h(x) \neq y \end{cases}.$$

Given any probability distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, the Bayes classifier $f_{\mathcal{D}}$ is defined by

$$f_{\mathcal{D}}(x) = \begin{cases} 1 & \text{if } \mathbb{P}[y = 1|x] \geq 1/2 \\ 0 & \text{otherwise} \end{cases}.$$

Show that for every probability distribution $\mathcal{D}$, the Bayes classifier $f_{\mathcal{D}}$ is optimal, in the sense that for every classifier $g : \mathcal{X} \to \{0,1\}$,

$$L_{\mathcal{D}}(f_{\mathcal{D}}) \leq L_{\mathcal{D}}(g) \ .$$

2 VC Dimension

- Let $\mathcal{X} = \{0,1\}^n$ and $\mathcal{Y} = \{0,1\}$, for each $I \subseteq [n]$ define the parity function:

  $$h_I(\mathbf{x}) = \Big(\sum_{i \in I} x_i\Big) \bmod 2.$$

  What is the VC-dimension of the class $\mathcal{H}_{parity} = \{h_I \mid I \subseteq [n]\}$? Prove your answer, you may use results we proved in class.

- Given an integer k, let $([a_i, b_i])_{i=1}^k$ be any set of k intervals on $\mathbb{R}$ and define their union $A = \cup_{i=1}^k [a_i, b_i]$. The hypothesis class $\mathcal{H}_{k-intervals}$ includes the functions:

  $$h_A(x) = \begin{cases} 0 & x \notin A \\ 1 & x \in A \end{cases},$$

  for all choices of k intervals. Find the VC-dimension of $\mathcal{H}_{k-intervals}$ and prove your answer. Show that if we let A be any finite union of intervals (i.e. k is unlimited), then the resulting class $\mathcal{H}_{intervals}$ has VC-dimension ∞.

- The hypothesis class of non-homogenous halfspaces over $\mathbb{R}^d$ is defined as

  $$HS_d = \Big\{ h_{\mathbf{w},b} : \mathbf{w} \in \mathbb{R}^d, \ b \in \mathbb{R} \Big\} \ ,$$

  where $h_{\mathbf{w},b}(\mathbf{x}) = \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right)$. Show that the VC-dimension of HS_d is $d+1$.

3 PAC Learnability

- Let $\mathcal{X}$ be a domain set, $\mathcal{Y} = \{\pm 1\}$ and $\mathcal{H}$ an arbitrary hypothesis class. Assume there is an algorithm $\mathcal{A}$ such that for any distribution $\mathcal{D}$ that is realizable by $\mathcal{H}$, it holds that:

  $$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}^{0-1}(\mathcal{A}(S))] \leq \mathbb{E}_{S \sim \mathcal{D}^m}[L_S^{0-1}(\mathcal{A}(S))] + \epsilon_m,$$

where $\epsilon_m \to 0$. Does this mean $\mathcal{H}$ is PAC-learnable? Explain your answer, you are not required to provide a formal proof.

- Let $\mathcal{H}$ be a hypothesis class of binary classifiers. Show that if $\mathcal{H}$ is agnostic PAC learnable, then $\mathcal{H}$ is PAC learnable as well. Furthermore, if A is a successful agnostic PAC learner for $\mathcal{H}$, then A is also a successful PAC learner for $\mathcal{H}$.

4 Multivariable Calculus

- For a fixed vector $\boldsymbol{x} \in \mathbb{R}^n$ and an orthogonal matrix $U \in \mathbb{R}^{n\times n}$, calculate the Jacobian of the function $f : \mathbb{R}^n \to \mathbb{R}^n$:

  $$f(\boldsymbol{\sigma}) = U\text{diag}(\boldsymbol{\sigma})U^T\boldsymbol{x}.$$

  Here $\text{diag}(\boldsymbol{\sigma})$ is an $n \times n$ matrix where $\text{diag}(\boldsymbol{\sigma})_{ij} = \begin{cases} \sigma_i & i = j \\ 0 & i \neq j \end{cases}$.

- Use the chain rule to calculate the gradient of $h(\boldsymbol{\sigma}) = \frac{1}{2}\|f(\boldsymbol{\sigma}) - y\|^2$

5 Visualizing Empirical Losses

Let $\mathcal{D}_x = U([0,1]^2)$, that is X_1, X_2 are two independent variables distributed uniformly on $[0,1]$, $\mathcal{Y} = \mathbb{R}$, and assume

$$\mathcal{D}_{y|x} = \langle w^*, x\rangle + \mathcal{N}(0, \epsilon),$$

where $w^* = (0.6, -1)$. Our hypothesis class is the class of homogenous linear separators of dimension 2, $\mathcal{H} = \{h_w(x) = \langle w, x\rangle \mid w \in \mathbb{R}^2\}$, the loss function we consider is the squared loss $l(h(x), y) = \frac{1}{2}\left(\langle w, x\rangle - y\right)^2$.

Write all your code for this question in a file called ex2.py.

- Write a python function sample that receives m, ϵ as input, draws m samples from $D_{x,y}$, puts them into a matrix $S \in \mathbb{R}^{3\times m}$ and returns S. The first column of S should contain values of x_1, the second of x_2 and the third of y. You may use the functions numpy.random.uniform, numpy.random.normal.

- The following lines generate matrices of vectors w that correspond to hypotheses h_w in our class:

```
w_low = -3
w_up = 3
mesh_range = np.arange(w_low,w_up,(w_up-w_low)/200)
ws = np.meshgrid(mesh_range,mesh_range)
```

Use them to write a function calcLS that receives a sample S and computes the matrix $L^S \in \mathbb{R}^{200\times 200}$, where L^S_{ij} is the empirical loss over S of the hypothesis $h_w(x)$ for the vector w = (ws[0][i,j], ws[1][i,j]).

- Write a function calcGradient that receives a sample S and vector w and outputs $\nabla L_S(w)$.

- Put the file gradDescent.py that can be found on the course website in the same directory as ex2.py. Use the previous functions to draw a sample $S_{5,1}$ with $m = 5, \epsilon = 1$ and a sample $S_{1000,0.001}$ with $m = 1000, \epsilon = 0.001$, calculate L^S for both these samples using the function you wrote.

- The function gradDescent in gradDescent.py recieves a sample S and its empirical loss L^S and plots:

  1. The empirical loss you provide it with (this is represented by colors).
  2. Iterations of a simple gradient descent procedure (as Shai showed you in class) over the empirical loss $L_S(h_w)$. These are represented as arrows from w_t to w_{t+1}.

  It also plots the sample S in a 3 dimensional scatter plot. Use this function with $S_{5,1}, L^{S_{5,1}}$ and $S_{1000,0.001}, L^{S_{1000,0,001}}$ and include the plots you get in your submission.

- Did the gradient descent procedure solve the ERM principle in both cases? Is the hypothesis found by gradient descent in both cases close to the best hypothesis h_{w^*}? Write your answers to these questions in the pdf file you submit.