COMS 4771 HW 2 (Spring 2020)

Due: Mar 08, 2020 at 11:59pm

You are allowed to write up solutions in groups of (at max) three students. These group members don't necessarily have to be the same from previous homeworks. Only one submission per group is required by the due date on Gradescope. Name and UNI of all group members must be clearly specified on the homework. No late homeworks are allowed. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible solutions for homework questions is encouraged on piazza and with peers outside your group, but every group must write their own individual solutions. You should cite all resources (including online material, books, articles, help taken from specific individuals, etc.) you used to complete your work.

1 A variant of Perceptron Algorithm

One is often interested in learning a *disjunction model*. That is, given binary observations from a d feature space (i.e. each datapoint $x \in \{0,1\}^d$), the output is an 'OR' of few of these features (e.g. $y = x_1 \vee x_{20} \vee x_{33}$). It turns out that the following variant of the perceptron algorithm can learn such disjunctions well.

Perceptron OR Variant
learning:
- Initialize $w := 1$
- for each datapoint (x, y) in the training dataset
- $\hat{y} := \mathbf{1}[w \cdot x > d]$
- if $y \neq \hat{y}$ and $y = 1$
- $w_i \leftarrow 2w_i$ (for $\forall i : x_i = 1$)
- if $y \neq \hat{y}$ and $y = 0$
- $w_i \leftarrow w_i/2$ (for $\forall i : x_i = 1$)

classification:
$f(x) := \mathbf{1}[w \cdot x > d]$

We will prove the following interesting result: The **Perceptron OR Variant** algorithm makes at most $2 + 3r(1 + \log d)$ mistakes when the target concept is an OR of r variables.

(i) Show that for any positive example, **Perceptron OR Variant** makes $M_+ \leq r(1 + \log d)$ mistakes.

(ii) Show that each mistake made on a negative example decreases the total weight $\sum_i w_i$ by at least $d/2$.

(iii) Let M_- denote the total number of mistakes on negative examples, and $TW(t)$ denote the total weight of w at iteration t. Observing that

$$0 < TW(t) \leq TW(0) + dM_+ - (d/2)M_-,$$

conclude that the total number of mistakes (i.e., $M_+ + M_-$) is at most $2 + 3r(1 + \log d)$.

2 Inconsistency of the fairness definitions

Recall the notation and definitions of group-based fairness conditions:
Notation:

Denote $X \in \mathbb{R}^d,\ A \in \{0,1\}$ and $Y \in \{0,1\}$ to be three random variables: non-sensitive features of an instance, the instance's sensitive feature and the target label of the instance respectively, such that $(X, A, Y) \sim \mathcal{D}$. Denote a classifier $f : \mathbb{R}^d \to \{0,1\}$ and denote $\hat{Y} := f(X)$.

For simplicity, we also use the following abbreviations:

$$\mathbb{P} := \mathbb{P}_{(X,A,Y)\sim D} \qquad \text{and} \qquad \mathbb{P}_a := \mathbb{P}_{(X,a,Y)\sim D}$$

Group based fairness definitions:
- Demographic Parity (DP)

$$\mathbb{P}_0[\hat{Y} = \hat{y}] = \mathbb{P}_1[\hat{Y} = \hat{y}] \qquad \forall \hat{y} \in \{0,1\}$$

(equal positive rate across the sensitive attribute)

- Equalized Odds (EO)

$$\mathbb{P}_0[\hat{Y} = \hat{y} \mid Y = y] = \mathbb{P}_1[\hat{Y} = \hat{y} \mid Y = y] \qquad \forall \hat{y},\ y \in \{0,1\}$$

(equal true positive- and true negative-rates across the sensitive attribute)

- Predictive Parity (PP)

$$\mathbb{P}_0[Y = y \mid \hat{Y} = \hat{y}] = \mathbb{P}_1[Y = y \mid \hat{Y} = \hat{y}] \qquad \forall \hat{y},\ y \in \{0,1\}$$

(equal positive predictive- and negative predictive-value across the sensitive attribute)

Unfortunately, achieving all three fairness conditions simultaneously is not possible. An impossibility theorem for group-based fairness is stated as follows.

- If A is dependent on Y, then Demographic Parity and Predictive Parity cannot hold at the same time.
- If A is dependent on Y and $\hat{Y}$ is dependent on Y, then Demographic Parity and Equalized Odds cannot hold at the same time.
- If A is dependent on Y, then Equalized Odds holds and Predictive Parity cannot hold at the same time.

These three results collectively show that it is impossible to simultaneously satisfy the fairness definitions except in some trivial cases.

(i) State a scenario where all three fairness definitions are satisfied simultaneously.

(ii) Prove the first statement.

(iii) Prove the second statement.

(iv) Prove the third statement.

Hint: First observe that

$$\mathbb{P}_0[Y = y|\hat{Y} = \hat{y}] = \mathbb{P}_1[Y = y|\hat{Y} = \hat{y}] \; \forall \hat{y}, \; y \in \{0, 1\}$$

is equivalent to:

$$\mathbb{P}_0[Y = 1|\hat{Y} = \hat{y}] = \mathbb{P}_1[Y = 1|\hat{Y} = \hat{y}] \; \forall \hat{y} \in \{0, 1\}.$$

A necessary condition for PP is the equality of positive predictive value (PPV):

$$\mathbb{P}_0[Y = 1|\hat{Y} = 1] = \mathbb{P}_1[Y = 1|\hat{Y} = 1]$$

To prove the third statement, it is enough to prove a stronger statement: if A is dependent on Y, Equalized Odds and equality of Positive Predictive Value cannot hold at the same time.

Next, try to express the relationship between FPR_a ($= \mathbb{P}_a[\hat{Y} = 1|Y = 0]$) and FNR_a ($= \mathbb{P}_a[\hat{Y} = 0|Y = 1]$) using p_a ($= \mathbb{P}[Y = 1 \mid A = a]$) and PPV_a ($= \mathbb{P}_a[Y = 1|\hat{Y} = 1]$), $\forall a \in \{0, 1\}$ and finish the proof.

3 Making data linearly separable by feature space mapping

Consider the infinite dimensional feature space mapping

$$\begin{aligned} &\Phi_\sigma : \mathbb{R} \to \mathbb{R}^\infty \\ &\qquad x \mapsto \Big(\mathbf{1}\big[|\alpha - x| < \sigma\big] \cdot \exp\big(-1/(1 - (|\alpha - x|/\sigma)^2)\big)\Big)_{\alpha \in \mathbb{R}}. \end{aligned}$$

(It may be helpful to sketch the function $f(\alpha) := \mathbf{1}\big[|\alpha| < 1\big] \cdot \exp\big(-1/(1-\alpha^2)\big)$ for understanding the mapping and answering the questions below)

(i) Show that for any n distinct points $x_1, \ldots, x_n$, there exists a $\sigma > 0$ such that the mapping Φ_σ can linearly separate *any* binary labeling of the n points.

(ii) Show that one can efficiently compute the dot products in this feature space, by giving an analytical formula for $\Phi_\sigma(x) \cdot \Phi_\sigma(x')$ for arbitrary points x and x'.

(iii) Given an input space X and a feature space mapping ϕ that maps elements from X to a (possibly infinite dimensional) inner product space V. Let $K : X \times X \to \mathbb{R}$ be a kernel function that can efficiently compute the inner products in V, that is, for any $x, x' \in X$, $K(x, x') = \phi(x) \cdot \phi(x')$.

Consider a binary classification algorithm that predicts the label of an unseen instance according to the class with the closest average. Formally, given a training set $S = (x_1, y_1), \ldots, (x_m, y_m)$, for each $y \in \{\pm 1\}$ define

$$c_y := \frac{1}{m_y} \sum_{i:y_i=y} \phi(x_i),$$

where $m_y = |\{i : y_i = y\}|$. Assume that m_+ and m_- are nonzero. Then, the algorithm outputs the following decision rule:

$$h(x) := \begin{cases} +1 & \|\phi(x) - c_+\| \leq \|\phi(x) - c_-\| \\ -1 & \text{otherwise.} \end{cases}$$

(a) Let $w := c_+ - c_-$ and let $b = \frac{1}{2}(\|c_-\|^2 - \|c_+\|^2)$. Show that

$$h(x) = \text{sign}(\langle w, \phi(x) \rangle + b).$$

(b) Show that $h(x)$ can be expressed via $K(\cdot, \cdot)$, without accessing individual entries of $\phi(x)$ or w, thus showing that $h(x)$ is efficiently computable.

4 Convexity

Let $f : \mathbb{R}^d \to \mathbb{R}$ be a twice differentiable function with a positive semidefinite hessian. For any two points $x, y \in \mathbb{R}^d$, we will show that the one-dimensional function $g : [0, 1] \to \mathbb{R}$ defined as

$$g(t) := f(tx + (1 - t)y)$$

is convex.

(i) Compute $g'(t)$ and $g''(t)$, and show that $g''(t) \geq 0$.

(ii) Using Taylor's theorem prove that

$$g(0) \geq g(t) + g'(t)(-t)$$
$$g(1) \geq g(t) + g'(t)(1 - t)$$

(iii) From part (ii), conclude that g is convex.

5 Empirical study of various gradient based optimization procedures

There are several gradient based optimization procedures that have been proposed over the years for finding local optima of a smooth non-convex function. A few of them are listed below.

- (Full) Gradient Descent (GD)
- Stochastic Gradient Descent (SGD)

- SGD with Momentum (SGDM)
- Adaptive Gradient (AdaGrad)
- RMSProp
- AdaDelta
- Adaptive Momentum (ADAM)

Using online resources, books, etc. your goal is to learn about what each technique is about and what makes each technique different. (make sure to properly cite all resources used!)

(i) Describe each of the techniques in detail. Compare and contrast the relative advantages and disadvantages of each of the techniques. Are there specific types of datasets where one technique is expected to perform better than the other? if so, why? Provide detailed justifications for each.

(ii) Implement each of the optimization procedures and experimentally verify your findings in part (i). Using synthetic dataset(s) compare and contrast the relative performance of each of the techniques.

You must submit your code to Courseworks to receive full credit.

6 Perceptron case study

We shall study the relative performance of different variations of the Perceptron algorithm on the handwritten digits dataset.

The dataset contains 10,000 images (each of size 28x28 pixels = 784 dimensions) of handwritten digits along with the associated labels. Each handwritten digit belongs to one of the 10 possible categories $\{0, 1, \ldots, 9\}$. There are two variables in this datafile: (i) Variable X is a 10,000x784 data matrix, where each row is a sample image of a handwritten digit. (ii) Variable Y is the 10,000x1 label vector where the i^{th} entry indicates the label of the i^{th} sample image in X.

Special note for those who are not using Matlab: Python users can use `scipy` to read in the mat file, R users can use `R.matlab` package to read in the mat file, Julia users can use `JuliaIO/MAT.jl`. Octave users should be able to load the file directly.

To visualize this data (in Matlab): say you want to see the actual handwritten character image of the 77th datasample. You may run the following code (after the data has been loaded):

```
figure;
imagesc(1-reshape(X(77,:),[28 28])');
colormap gray;
```

To see the associated label value:

```
Y(77)
```

Consider a sequence of training data $(x_1, y_1), \ldots, (x_n, y_n)$ in an arbitrary but fixed order (the labels y_i are assumed to be binary in $\{-1, +1\}$).

Perceptron V0
learning:
- Initialize $w_0 := 0$
- for $t = 1, \ldots, T$
- pick example (x_i, y_i), where $i = (t \bmod n + 1)$
- if $y_i(w_{t-1} \cdot x_i) \leq 0$
- $w_t := w_{t-1} + y_i x_i$
- else
- $w_t := w_{t-1}$

classification:
$f(x) := \text{sign}(w_T \cdot x)$

Perceptron V1
learning:
- Initialize $w_0 := 0$
- for $t = 1, \ldots, T$
- pick example (x_i, y_i), such that $i := \arg\min_j \left(y_j w_{t-1} \cdot x_j\right)$
- if $y_i(w_{t-1} \cdot x_i) \leq 0$
- $w_t := w_{t-1} + y_i x_i$
- else
- $w_T := w_{t-1}$; terminate

classification:
$f(x) := \text{sign}(w_T \cdot x)$

Perceptron V2
learning:
- Initialize $w_1 := 0$, $c_0 := 0$, k := 1
- for $t = 1, \ldots, T$
- pick example (x_i, y_i), where $i = (t \bmod n + 1)$
- if $y_i(w_k \cdot x_i) \leq 0$
- $w_{k+1} := w_k + y_i x_i$
- $c_{k+1} := 1$
- $k := k + 1$
- else
- $c_k := c_k + 1$

classification:
$f(x) := \text{sign}\big(\sum_{i=1}^{k} c_i \ \text{sign}(w_i \cdot x)\big)$

(i) Implement the three variations of the Perceptron algorithm for the 10-way digit classification problem.
You must submit your code on Courseworks to receive full credit.

(ii) Which Perceptron version is better for classification? You must justify your answer with appropriate performance graphs demonstrating the superiority of one classifier over the other. Example things to consider: you should evaluate how the classifier behaves on a holdout

'test' sample for various splits of the data; how does the training sample size and the number of passes affects the classification performance.

(iii) Implement the Kernel Perceptron as described in lecture with a high degree (say, 5 to 10) polynomial kernel. How does it affect the classification on test data?