CS480/680: Introduction to Machine Learning

Homework 1

Due: 11:59 pm, Jan 30, 2025, submit on Crowdmark.

NAME

student number

Submit your writeup in pdf and all source code in a zip file (with proper documentation). Write a script for each programming exercise so that the TA can easily run and verify your results. Make sure your code runs!
[Text in square brackets are hints that can be ignored.]

Exercise 1: Perceptron (8 pts)

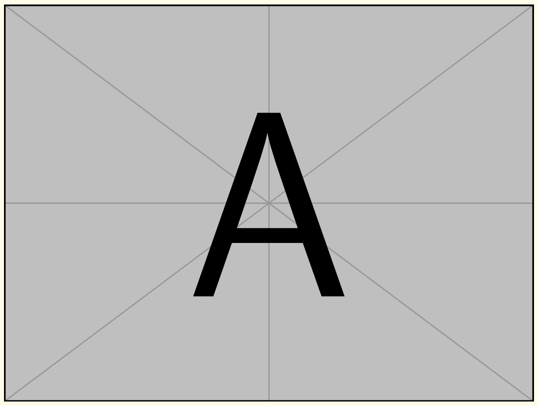
Convention: All algebraic operations, when applied to a vector or matrix, are understood to be element-wise (unless otherwise stated).

Algorithm 1: The perceptron.

```
Input: $X \in \mathbb{R}^{d\times n}$, $\mathbf{y} \in \{-1,1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, max_pass $\in \mathbb{N}$
Output: $\mathbf{w}, b, mistake$
1 for $t = 1, 2, \ldots,$ max_pass do
2     $mistake(t) \leftarrow 0$
3     for $i = 1, 2, \ldots, n$ do
4         if $y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \le 0$ then
5             $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$          // $\mathbf{x}_i$ is the $i$-th column of $X$
6             $b \leftarrow b + y_i$
7             $mistake(t) \leftarrow mistake(t) + 1$
```

1. (2 pts) Implement the perceptron in Algorithm 1. Your implementation should take input as $X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, $\mathbf{y} \in \{-1,1\}^n$, an initialization of the hyperplane parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and the maximum number of passes of the training set [suggested max_pass = 500]. Run your perceptron algorithm on the spambase dataset (available on course website), and plot the number of mistakes (y-axis) w.r.t. the number of passes (x-axis).

   Ans:

2. (2 pts) Consider the (continuous) piece-wise function

$$f(\mathbf{w}) := \max_k f_k(\mathbf{w}), \tag{1}$$

   where each f_k is continuously differentiable. We define the derivative of f at any $\mathbf{w}$ as follows: first find (any) k such that $f(\mathbf{w}) = f_k(\mathbf{w})$, i.e., $f_k(\mathbf{w})$ achieves the maximum among all pieces; then we let $f'(\mathbf{w}) = f'_k(\mathbf{w})$. [Clearly, the index k that achieves maximum may depend on $\mathbf{w}$, the point we evaluate the derivative at.] Now consider the following problem [padding applied, $y_i \in \{\pm 1\}$]:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \max\{0, -y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle)\}. \tag{2}$$

Prove that in each iteration, the (binary) perceptron algorithm essentially picks a term from the above summation, computes the corresponding derivative (say $\mathbf{g}$), and performs a gradient update:

$$\mathbf{w} \leftarrow \mathbf{w} - \mathbf{g}. \tag{3}$$

[You may ignore the degenerate case when $\langle \mathbf{x}_i, \mathbf{w} \rangle = 0$, and you can use the usual chain rule for our derivative.]

Ans:

3. (2 pts) Consider the following problem, where $y_i \in \{1, 2, \ldots, c\}$:

$$\min_{\mathbf{w}_1, \ldots, \mathbf{w}_c} \sum_{i=1}^{n} \max_{k=1,\ldots,c} \Big[\langle \mathbf{x}_i, \mathbf{w}_k \rangle - \langle \mathbf{x}_i, \mathbf{w}_{y_i} \rangle \Big]. \tag{4}$$

Show that when $c = 2$, we reduce to the binary perceptron problem in (2). [Try to identify the weights $\mathbf{w}$, using some transformation.]

Ans:

4. (2 pts) Based on the analogy to the binary case, develop and implement a multiclass perceptron algorithm to solve (4) directly. Run your implementation on the activity dataset (available on course website) and report the final errors on the training and test sets. [Hint: obviously, we want to predict as follows: $\hat{y} = \operatorname*{argmax}_{k=1,\ldots,c} \langle \mathbf{x}, \mathbf{w}_k \rangle$, i.e., the class k whose corresponding $\mathbf{w}_k$ maximizes the inner product. Explain your algorithm (e.g., through pseudo-code).]

Ans:

Exercise 2: Perceptron on non-separable dataset (6 pts)

In this exercise we develop a mistake bound for perceptron on even a non-separable dataset. For any normal vector $\mathbf{w}$ and scalar $s > 0$, define the margin on the i-th training example (assuming padding is applied):

$$m_i = \max\{0, s - y_i\langle \mathbf{x}_i, \mathbf{w}\rangle\}. \tag{5}$$

For this exercise we only consider a single pass of Algorithm 1 (i.e., $\mathsf{max\_pass} = 1$).

Our goal is to prove that

$$\mathit{mistake} := \mathit{mistake}(1) \le \left(\frac{R\|\mathbf{w}\|_2 + \|\mathbf{m}\|_2}{s}\right)^2, \tag{6}$$

where recall that $R := \max_i \|\mathbf{x}_i\|_2$ and the vector $\mathbf{m} \in \mathbb{R}^n$ is defined in (5).

1. (1 pt) We construct a new dataset $\tilde{\mathbf{x}}_i \in \mathbb{R}^{d+n}, i = 1, \ldots, n$ such that

$$\tilde{\mathbf{x}}_i = \mathtt{concatenate}(\mathbf{x}_i, c \cdot \mathbf{e}_i), \tag{7}$$

where $\mathbf{e}_i$ is the i-th standard basis in $\mathbb{R}^n$ (i.e., with 1 at the i-th entry and 0 elsewhere) and $c > 0$ is a constant that we determine later. We keep the label y_i the same as before. Prove that the new dataset is separable. In particular, consider the hyperplane

$$\tilde{\mathbf{w}} = \mathtt{concatenate}(\mathbf{w}, \mathbf{y} \odot \mathbf{m}/c) \tag{8}$$

and show that for all i,

$$y_i\langle \tilde{\mathbf{x}}_i, \tilde{\mathbf{w}}\rangle \ge s > 0. \tag{9}$$

[We remind that $\odot$ is the Hadamard product: if $\mathbf{c} = \mathbf{a} \odot \mathbf{b}$ then $c_i = a_i b_i$ for all i.]

Ans:

2. (2 pts) Apply the perceptron convergence guarantee (on slide 22, with $\delta = 0$ and $\mathbf{w}$ initialized as $\mathbf{0}$) to the above separable dataset $\{\tilde{\mathbf{x}}_i, y_i\}$ to bound the mistake:

$$\mathit{mistake} = \mathit{mistake}(1) \le \frac{(R\|\mathbf{w}\|_2 + \|\mathbf{m}\|_2)^2}{s^2}. \tag{10}$$

[You need to decide the value of c.]

Ans:

3. (3 pts) Prove that the perceptron makes the same mistakes on the dataset $\{(\mathbf{x}_i, y_i)\}$ and the new dataset $\{(\tilde{\mathbf{x}}_i, y_i)\}$. This completes our proof for a mistake bound of perceptron on a (possibly) non-separable dataset. We remind that we only consider $\mathsf{max\_pass} = 1$ and we pick the data points sequentially.

[Hint: if $\mathbf{w}$ is the initialization of perceptron on the dataset $\{(\mathbf{x}_i, y_i)\}$, use $\tilde{\mathbf{w}} := \texttt{concatenate}(\mathbf{w}, \mathbf{0})$ on the dataset $\{(\tilde{\mathbf{x}}_i, y_i)\}$. Apply induction. If get stuck, can run your implementation on the two datasets side by side to get further insights.]

Ans:

Exercise 3: Generalized linear models (6 pts)

Recall that in logistic regression we assumed the *binary* label $\mathsf{Y}_i \in \{0, 1\}$ follows the Bernoulli distribution: $\Pr(\mathsf{Y}_i = 1|\mathsf{X}_i) = p_i$, where p_i also happens to be the mean. Under the independence assumption we derived the (conditional) negative log-likelihood function:

$$-\sum_{i=1}^{n}(1-y_i)\log(1-p_i) + y_i \log(p_i). \tag{11}$$

Then, we parameterized the mean parameter p_i through the logit transform:

$$\log\frac{p_i}{1-p_i} = \langle \mathbf{x}_i, \mathbf{w}\rangle + b, \quad \text{or equivalently} \quad p_i = \frac{1}{1+\exp(-\langle \mathbf{x}_i, \mathbf{w}\rangle - b)}. \tag{12}$$

Lastly, we found the weight vector $\mathbf{w}$ and b by minimizing the negative log-likelihood function.

In the following we generalize the above idea significantly. Let the (conditional) density of Y (given $\mathsf{X} = \mathbf{x}$) be

$$p(y|\mathbf{x}) = \exp\Big[\mu(\mathbf{x})\cdot y - \lambda(\mathbf{x})\Big]\cdot q(y), \tag{13}$$

where $\mu : \mathbb{R}^d \to \mathbb{R}$ is a function of $\mathbf{x}$ and $\lambda(\mathbf{x}) = \log\int_y \exp\big(\mu(\mathbf{x})\cdot y\big)q(y)\mathrm{d}y$ so that $p(y|\mathbf{x})$ is properly normalized wrt y (i.e., integrate to 1). For discrete y (such as in logistic regression), replace the density with the probability mass function and the integral with sum.

As always, you need to supply sufficient derivation details to justify your final answer.

1. (1 pt) Given a dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, derive the (conditional) negative log-likelihood function of $y_1, \ldots, y_n$, assuming independence and the density form in (13).

   Ans: We have

$$TBD \tag{14}$$

2. (1 pt) Plug the usual linear parameterization

$$\mu(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w}\rangle + b = \langle \mathbf{x}, \mathbf{w}\rangle \tag{15}$$

   into your (conditional) negative log-likelihood and compute the gradient of the resulting function. [Hint: you may swap differentiation with integral and your gradient may involve implicitly defined terms.]

Ans: We have

$$\ell_n(\mathbf{w}) = \tag{16}$$

and hence

$$\nabla\ell_n(\mathbf{w}) = \tag{17}$$

3. (1 pt) Let us revisit linear regression, where

$$p(y|\mathbf{x}) = \tfrac{1}{\sqrt{2\pi}} \exp\left(- \tfrac{(y-\nu(\mathbf{x}))^2}{2}\right) \tag{18}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 3.2, derive the negative log-likelihood and gradient. [Hint: you may simply plug into the more general result in Ex 3.2. Compare with what you already learned about linear regression to make sure both Ex 3.2 and Ex 3.3 are correct.]

Ans: We have

$$\mu(\mathbf{x}) = \tag{19}$$
$$\lambda(\mathbf{x}) = \tag{20}$$
$$q(y) = \tag{21}$$
$$\ell_n(\mathbf{w}) = \tag{22}$$
$$\nabla\ell_n(\mathbf{w}) = \tag{23}$$

4. (1 pt) Let us revisit logistic regression, where

$$\Pr(\mathsf{Y} = y|\mathbf{x}) = [\nu(\mathbf{x})]^y[1-\nu(\mathbf{x})]^{1-y}, \quad \text{where} \quad y \in \{0,1\}. \tag{24}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 3.2, derive the negative log-likelihood and gradient. [Hint: Compare with what you already learned about logistic regression.]

Ans: We have

$$\mu(\mathbf{x}) = \tag{25}$$
$$\lambda(\mathbf{x}) = \tag{26}$$
$$q(y) = \tag{27}$$
$$\ell_n(\mathbf{w}) = \tag{28}$$
$$\nabla\ell_n(\mathbf{w}) = \tag{29}$$

5. (2 pts) Now let us tackle something new. Let

$$\Pr(\mathsf{Y} = y|\mathbf{x}) = \frac{[\nu(\mathbf{x})]^y}{y!}\exp(-\nu(\mathbf{x})), \quad \text{where} \quad y = 0,1,2,\ldots. \tag{30}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 3.2, derive the negative log-likelihood and gradient. [Hint: Y here follows the Poisson distribution, which is useful for modeling integer-valued events, e.g., the number of customers at a given time.]

Ans: We have

$$\mu(\mathbf{x}) = \tag{31}$$
$$\lambda(\mathbf{x}) = \tag{32}$$
$$q(y) = \tag{33}$$
$$\ell_n(\mathbf{w}) = \tag{34}$$
$$\nabla\ell_n(\mathbf{w}) = \tag{35}$$