# CS480/680: Introduction to Machine Learning

Homework 1

Due: 11:59 pm, October 04, 2022, submit on LEARN.

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).

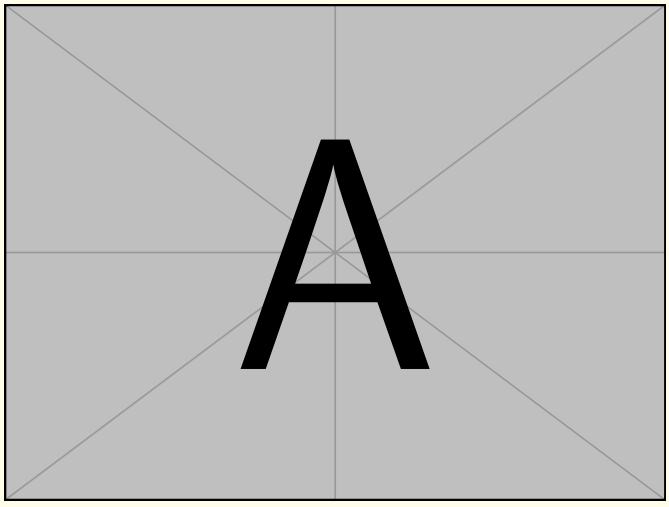
**Algorithm 1:** The perceptron.

**Input:** $X \in \mathbb{R}^{d \times n}$, $\mathbf{y} \in \{-1, 1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, $\mathsf{max\_pass} \in \mathbb{N}$
**Output:** $\mathbf{w}, b, mistake$

```
for t = 1, 2, ..., max_pass do
    mistake(t) ← 0
    for i = 1, 2, ..., n do
        if y_i(⟨x_i, w⟩ + b) ≤ 0 then
            w ← w + y_i x_i                    // x_i is the i-th column of X
            b ← b + y_i
            mistake(t) ← 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 $\mathsf{max\_pass} = 500$]. 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. (1 pt) Using the one-vs-all reduction to implement a multiclass perceptron. You may call your binary implementation. Test your algorithm on the activity dataset (available on course website), and report your final errors on the training and test sets.

   Ans:

3. (1 pt) Using the one-vs-one reduction to implement a multiclass perceptron. You may call your binary implementation. Test your algorithm on the activity dataset (available on course website), and report your final errors on the training and test sets.

   Ans:

4. (1 pt) 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 its derivative 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:

5. (1 pt) 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).

   Ans:

6. (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: $\hat{y} = \operatorname{argmax}_{k=1,\dots,c} \langle \mathbf{x}, \mathbf{w}_k \rangle$, i.e., the class $k$ whose corresponding $\mathbf{w}_k$ maximizes the inner product. Giving the correct pseudo-code will receive 1 pt.]

   Ans:

## Exercise 2: 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:

$$-\sum_{i=1}^{n} (1 - y_i) \log(1 - p_i) + y_i \log(p_i). \tag{5}$$

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

$$\log \frac{p_i}{1 - p_i} = \langle \mathbf{x}_i, \mathbf{w} \rangle + b, \quad \text{or equivalently} \quad p_i = \frac{1}{1 + \exp(-\langle \mathbf{x}_i, \mathbf{w} \rangle - b)}. \tag{6}$$

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 $\mathsf{Y}$ (given $\mathsf{X} = \mathbf{x}$) be

$$p(y|\mathbf{x}) = \exp\Big[\mu(\mathbf{x}) \cdot y - \lambda(\mathbf{x})\Big] \cdot q(y), \tag{7}$$

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 $y_1, \dots, y_n$, assuming independence.

   Ans: We have

$$TBD \tag{8}$$

2. (1 pt) Plug the usual linear parameterization

$$\mu(\mathbf{x}) = \langle \mathbf{x}, \mathbf{w} \rangle + b = \langle \mathbf{x}, \mathbf{w} \rangle \tag{9}$$

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

$$\ell_n(\mathbf{w}) = \tag{10}$$

and hence

$$\nabla \ell_n(\mathbf{w}) = \tag{11}$$

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

$$p(y|\mathbf{x}) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y-\nu(\mathbf{x}))^2}{2}\right) \tag{12}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 2.2, derive the negative log-likelihood and gradient. [Hint: you may simply plug into the more general result in Ex 2.2. Compare with what you already learned about linear regression to make sure both Ex 2.2 and Ex 2.3 are correct.]

Ans: We have

$$\mu(\mathbf{x}) = \tag{13}$$
$$\lambda(\mathbf{x}) = \tag{14}$$
$$q(y) = \tag{15}$$
$$\ell_n(\mathbf{w}) = \tag{16}$$
$$\nabla \ell_n(\mathbf{w}) = \tag{17}$$

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

$$\Pr(\mathsf{Y} = y|\mathbf{x}) = [\nu(\mathbf{x})]^y [1-\nu(\mathbf{x})]^{1-y}, \quad \text{where} \quad y \in \{0, 1\}. \tag{18}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 2.2, derive the negative log-likelihood and gradient. [Hint: Compare with what you already learned about logistic regression.]

Ans: We have

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

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

$$\Pr(\mathsf{Y} = y|\mathbf{x}) = \frac{[\nu(\mathbf{x})]^y}{y!} \exp(-\nu(\mathbf{x})), \quad \text{where} \quad y = 0, 1, 2, \ldots. \tag{24}$$

Identify the functions $\mu(\mathbf{x})$, $\lambda(\mathbf{x})$ and $q(y)$ for the above specialization. Based on the linear parameterization in Ex 2.2, derive the negative log-likelihood and gradient. [Hint: $\mathsf{Y}$ 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

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

## Exercise 3: Ordinal regression (6 pts)

In many applications, the "labels" have an inherent order. For example, the letter grade $A$ is preferred to $B$, which is preferred to $C$, etc. More generally, consider $c$ ordinal labels $1, 2, \ldots, c$, where we prefer label $k$ than $k+1$, for each $k = 1, \ldots, c-1$. [The preference is transitive, i.e. any "smaller" label is preferred than a "larger" label.]

1. (1 pt) We could simply ignore the ordering information in the labels and treat the problem as an instance of multi-class classification. Can you name one possible downside of this approach?

   Ans:

2. (1 pt) We could also simply treat the labels as real numbers (which enjoy a natural ordering themselves), and hence treat the problem as an instance of regression. Can you name one possible downside of this approach? [Hint: it is often easy to decide if we prefer apples than oranges, but much harder to quantify how much we prefer apples than oranges.]

   Ans:

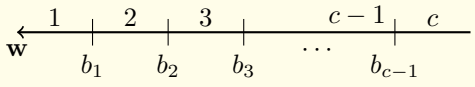
3. (2 pts) Let us consider $c-1$ *parallel* hyperplanes $H_k := \{\mathbf{x} : \langle \mathbf{x}, \mathbf{w} \rangle + b_k = 0\}$, which partition our space into $c$ rectangular regions. We define our prediction as

$$\hat{y} \leq k \iff \langle \mathbf{x}, \mathbf{w} \rangle + b_k > 0, \tag{30}$$

   or more explicitly,

$$\hat{y} = k \iff [\langle \mathbf{x}, \mathbf{w} \rangle + b_k > 0 \text{ and } \langle \mathbf{x}, \mathbf{w} \rangle + b_{k-1} \leq 0], \tag{31}$$

   where $b_0 := -\infty$ and $b_c := \infty$.




   The ordering in the labels is now respected, if we constrain $b_1 \leq b_2 \leq \cdots \leq b_{c-1}$:

$$\hat{y} \leq k \implies \hat{y} \leq l, \quad \forall\, l \geq k. \tag{32}$$

We learn the weights $\mathbf{w}$ and $b_1, \ldots, b_{c-1}$ by reducing to a sequence of (coupled) binary classifications:

$$\min_{\mathbf{w}, b_1 \le b_2 \le \cdots \le b_{c-1}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \sum_{k=1}^{c-1}\sum_{i=1}^{n} \max\{0, 1 - (\llbracket y_i = k \rrbracket - \llbracket y_i = k+1 \rrbracket)(\langle \mathbf{x}_i, \mathbf{w}\rangle + b_k)\}, \tag{33}$$

where recall that $\llbracket A \rrbracket$ is 1 if $A$ is true and 0 otherwise. It is clear that when $c = 2$, the above reduces to the familiar soft-margin SVM. Derive the Lagrangian dual of (33). [If it helps, you may ignore the constraint $b_1 \le \ldots \le b_{c-1}$.]

Ans:

4. (2 pts) In the previous formulation, to learn $b_k$, essentially we take class $k$ as positive and class $k+1$ as negative. Can you find a "better" alternative? Write down the formulation. [Hint: it would be similar to (33).]

   Ans: