STAT 154: Homework 7

Release date: **Sunday, April 21**

Due by: **11 PM, Friday, May 10**

The honor code

(a) Please state the names of people who you worked with for this homework. You can also provide your comments about the homework here.

(b) Please type/write the following sentences yourself and sign at the end. We want to make it *extra* clear that nobody cheats even unintentionally.

I hereby state that all of my solutions were entirely in my words and were written by me. I have not looked at another students solutions and I have fairly credited all external sources in this write up.

Submission instructions

- It is a good idea to revisit your notes, slides and reading; and synthesize their main points BEFORE doing the homework.
- You need to submit the following:
  1. A pdf of your write-up to "HW7 write-up".
  2. No *separate* code submission is required for this HW.
- Although, we have provided an extra week, we recommend you to finish the homework in two weeks so as to have enough time for revision.

Homework Overview

This homework covers support vector machines (SVM), logistic regression and decision trees.

1 True or False (8 pts)

State whether the given statements are True or False and provide a brief justification.

1. Logistic regression assumes that each class's points are generated from a Gaussian distribution.
2. Logistic regression cannot be kernelized.
3. Logistic Regression can be used for regression (when the response y take continuous value in $\mathbb{R}$).
4. Hard-margin SVM still return a classifier if the training data is not linearly separable.
5. In hard-margin SVM, support vectors are the only data points necessary to compute the fit $f(x) = w^\top x + b$.
6. In a soft-margin support vector machine, if we increase C (same notation as in note **n20_svm.pdf**), there will be more points inside the margin (i.e. between the two margin-defining hyperplanes).
7. As C approaches 0, the soft margin SVM solution approaches that of the hard margin SVM.
8. Recall that in the usual formulation of soft-margin SVMs, each training sample has a slack variable $\xi_i \geq 0$ and we impose a regularization term $C\sum_{i=1}^n \xi_i$. Consider an alternative formulation where we impose the additional constraints $\xi_i = \xi_j$ for all $(i, j) \in \{1, \ldots, n\}^2$. The minimum objective value $\|w\|_2^2 + C\sum_{i=1}^n \xi_i$ of the new formulation will be smaller than that of the original soft-margin SVM.

2 SVM on simple data (6 pts)

We would like to understand how SVM works on a trivial dataset of four data points in $\mathbb{R}^2$:

$$x_1 = \begin{bmatrix}1\\1\end{bmatrix}, \quad x_2 = \begin{bmatrix}1\\5\end{bmatrix}, \quad x_3 = \begin{bmatrix}5\\1\end{bmatrix}, \quad x_4 = \begin{bmatrix}5\\5\end{bmatrix}$$
$$y_1 = -1, \quad y_2 = 1, \quad y_3 = 1, \quad y_4 = 1$$

Recall from the lecture that in SVM, we want find a maximal-margin separating hyperplane of the form

$$f(x) = x^\top w + b$$

1. (1 pt) **Use pen and paper to draw the location of the maximal margin separating hyperplane for the given dataset and its two margin-defining hyperplanes.**

2. (1 pt) By visual examination, **guess the coordinates of the weight vector w (up to a constant).**

3. (2 pts) Next, **find the length of the normal** $\|w\|_2$, such that would assure that the closest training points are located at the margin-defining hyperplanes ± 1. For this, note that the distance between those two margin-defining hyperplanes is equal to $d = 2/\|w\|_2$.

4. (2 pts) **Rescale the w you guessed in Step 2 to have length you computed in Step 3 and then compute the corresponding intercept b by using the fact that $f(x_1) = -1$.**

3 Perceptron and Pegasos (17 pts)

Given a **convex** function $f : \mathbb{R}^d \to \mathbb{R}$, a vector $g_w \in \mathbb{R}^d$ is said to be the subgradient of f at w, if

$$f(z) \geq f(w) + g_w^\top (z - w) \quad \text{for all } z \in \mathbb{R}^d. \tag{1}$$

When the function is differentiable at w, the sub-gradient g_w is unique and is equal to the gradient, i.e., $g_w = \nabla_w f(w)$. In simple words, sub-gradient is a generalized gradient (to do math with non-differentiable functions). There could be $0, 1$, or infinitely many subgradients at any point. The sub-differential of f at a point $w$is the set of all subgradients of f at w and is denoted by $\partial f(w)$, .[1]

1. (2 pts) [Subgradients for point-wise maximum of functions] Suppose $f_1, \ldots, f_m : \mathbb{R}^d \to \mathbb{R}$ are convex functions, and

$$f(w) = \max_{i=1,\ldots,m} f_i(w).$$

   Let k be any index for which $f_k(w) = f(w)$, and choose $g_w \in \partial f_k(w)$. [We are using the fact that a convex function on $\mathbb{R}^d$ has a non-empty sub-differential at all points.] **Show using the definition (1) that g_w is a valid subgradient of f at w, i.e., $g_w \in \partial f(w)$.**

2. (2 pts) [Subgradient of hinge loss for linear prediction] **Find a subgradient** g_w of the function $f : \mathbb{R}^d \to \mathbb{R}$ where

$$f(w) = \max\left\{0, 1 - yw^\top x\right\},$$

   with $x \in \mathbb{R}^d$ and $y \in \{-1, 1\}$ as fixed.

We now work on deriving the perceptron algorithm, which is one of the first classification algorithms in machine learning. Suppose we have a labeled training set $\left\{(x_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\right\}_{i=1}^n$. In the perceptron algorithm, we are looking for a hyperplane that perfectly separates the two classes, i.e., we try to find a vector $w \in \mathbb{R}^d$ and an intercept $b \in \mathbb{R}$ such that

[1] In case you are more curious, see https://stanford.edu/class/ee364b/lectures/subgradients_notes.pdf by Boyd *et al.* for more details.

Algorithm 1: Perceptron Algorithm

Input: Training set $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{-1, 1\}$
Initialize $w^{(0)}$, $k = 0$;
while *allCorrect == FALSE* **do**
  allCorrect = TRUE;
  // Loop over all data points
  for $i = 1, 2, \ldots, n$ **do**
    if $y_i x_i^\top w^{(k)} \leq 0$ **then**
      $w^{(k+1)} = w^{(k)} + y_i x_i$;
      allCorrect = FALSE
    else
      $w^{(k+1)} = w^{(k)}$
    end
    $k = k + 1$;
  end
end
Output: $w^{(k)}$

$$w^\top x + b \begin{cases} > 0 & \text{if } y_i = 1 \\ < 0 & \text{if } y_i = -1 \end{cases} \iff y_i(w^\top x + b) > 0 \quad \forall i \in \{1, \ldots, n\} \tag{2}$$

For simplicity, in the following questions, we are leaving off the intercept term b.[2] Visually, all the data points with label $y = 1$ will lie on one side of the hyperplance and all the data points with label $y = -1$ will lie on the other side. When such a hyperplane exists, we say that the data are linearly separable. The perceptron algorithm is described in Algorithm 1

Give an estimator $\hat{y}$ for a label y, we define the **perceptron loss** as

$$\ell(\hat{y}, y) = \max\{0, -\hat{y}y\}. \tag{3}$$

3. (2 pts) **Discuss why the perceptron loss (3) is meaningful and would capture mismatch between $\hat{y}$ and y.** Define the average perceptron loss of w on the training set $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, as

$$\mathcal{L}(w; \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \ell(x_i^\top w, y_i).$$

Show that if $\{x | \widehat{w}^\top x = 0\}$ is a separating hyperplane (2) for the training set $\mathcal{D}$ then the average perceptron loss $\mathcal{L}(\widehat{w}; \mathcal{D})$ is 0. In other words, any separating hyperplane of $\mathcal{D}$ minimizes the average perceptron loss on the training set.

[2] Equivalently, we can assume that our feature vectors $\widetilde{x} = [x^\top, 1]^\top \in \mathbb{R}^{d+1}$ have a constant term 1 and we are looking to optimize over $\widetilde{w}^\top = [w^\top, b]^\top \in \mathbb{R}^{d+1}$ such that $y_i(\widetilde{w}^\top \widetilde{x}) > 0$.

4. (4 pts) Consider running stochastic subgradient descent (SSGD, subgradient descent on the loss for one data point at a time) to minimize the average perceptron loss. **Show that the perceptron algorithm is doing exactly SSGD with a constant step size** 1. We terminate when our training data are separated, we cycle through the data points one by one and we make the right choice of subgradient.
Hint: If g_k is the subgradient to be used at k-th step, then the update looks like

$$w^{(k+1)} = w^{(k)} - \alpha g_k,$$

where α is the step-size. To derive the sub-gradient, use results from questions 2.1 and 2.2.

5. (2 pts) Suppose the perceptron algorithm returns $\widehat{w}$. **Show that w is a linear combination of the input points, i.e., we can write $\widehat{w} = \sum_{i=1}^{n} \alpha_i x_i$ for some $\alpha \in \mathbb{R}^n$.**

Next, we try to solve SVM using the Pegasos algorithm (Pegasos: Primal Estimated sub-GrAdient SOlver for SVM). To be consistent with the paper, we consider the following ℓ_2-regularized formulation of the SVM objective

$$\min_{w \in \mathbb{R}^d} \mathcal{J}(w) \quad \text{where} J(w) \frac{\lambda}{2} \|w\|_2^2 + \frac{1}{n} \sum_{i=1}^{n} \max\left\{0, 1 - y_i w^\top x_i\right\}.$$

Again the intercept term is left out for simplicity. The Pegasos algorithm works as follows.

Algorithm 2: Pegasos Algorithm for SVM

Input: Training set $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{-1, 1\}$
Initialize $w^{(0)}$, $k = 0$;
while *not converged* **do**
  `// Loop over all data points`
  for $i = 1, 2, \ldots, n$ **do**
    $k = k + 1$;
    $\eta = 1/(k\lambda)$;
    if $y_i x_i^\top w^{(k)} < 1$ **then**
      $w^{(k+1)} = (1 - \eta\lambda) w^{(k)} + \eta y_i x_i$;
    else
      $w^{(k+1)} = (1 - \eta\lambda) w^{(k)}$
    end
  end
end
Output: $w^{(k)}$

6. (2 pts) Note that $\mathcal{J}(w) = \frac{1}{n} \sum_{i=1}^{n} J_i(w)$ where $J_i(w)$ denotes the SVM objective on a single data point:

$$J_i(w) = \frac{\lambda}{2} \|w\|_2^2 + \max\left\{0, 1 - y_i w^\top x_i\right\}.$$

Using the fact that J_i is a maximum of two convex functions, **show that a subgradient of J_i with respect to w is given by**

$$g_w = \begin{cases} \lambda w - y_i x_i & \text{for } y_i w^\top x_i < 1 \\ \lambda w & \text{for } y_i w^\top x_i \geq 1. \end{cases}$$

Hint: Use results from questions 2.1 and 2.2.

7. (3 pts) **Explain why the Pegasus algorithm is just stochastic subgradient descent (SSGD) on the regularized SVM loss $\mathcal{J}$ with decreasing step-size $\eta = 1/(k\lambda)$.**
Hint: Use the hints from 2.4.

4 Logistic Regression (10 pts)

We now work through some key steps for the IRWLS algorithm used for solving logistic regression. Given a dataset $\mathcal{D} = \left\{(x_i, y_i) \in \mathbb{R}^d \times \{0,1\}\right\}_{i=1}^n$, (2-class) logistic regression maximizes the conditional probability of the labels $\mathbb{P}(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta)$ assuming that (a) the data points are independent and that (b) for one data point, the conditional probability takes the form:

$$\mathbb{P}(Y_i = y_i | x_i; \beta) = \frac{e^{y_i \beta^\top x}}{1 + e^{\beta^\top x}}, \quad \text{for } y_i \in \{0,1\}. \tag{4}$$

1. (3 pts) **Derive the expression for the logistic loss as a function of β, which is defined as the negative conditional log-likelihood of the data:**

$$\mathcal{L}(\beta) = -\log \mathbb{P}(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta)$$

Also derive the expression for its gradient $\nabla_\beta \mathcal{L}(\beta)$ and the Hessian $\nabla^2_\beta \mathcal{L}(\beta)$. You may use the shorthand notation $p_i = \mathbb{P}(y_i = 1 | x_i; \beta)$.

2. (2 pts) **Show that the logistic loss $\beta \to \mathcal{L}(\beta)$ is convex.**
Hint: You may have to simply show some property for the Hessian.

3. (3 pts) Newton's method is effective for minimizing a convex function or maximizing a concave function. **Given an iterate β, derive the Newton's updates for the logistic loss**. Use the vector matrix notation:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} \in \mathbb{R}^{n \times d}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \in \{0,1\}^n, \qquad \mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix} \in [0,1]^n,$$

$$\mathbf{W} = \begin{bmatrix} p_1(1-p_1) & & & \\ & p_2(1-p_2) & & \\ & & \ddots & \\ & & & p_n(1-p_n) \end{bmatrix} \in \mathbb{R}^{n \times n}.$$

where $p_i = \mathbb{P}(y_i = 1 | x_i; \beta)$.

4. (2 pts) **Justify why this method is called iteratively re-weighted least squares (IRWLS/IRLS) for short.**

5 Decision Trees and Ada-boost (9 pts)

1. (3 pts) ISL Book: Problem 8.1 (Page 332)
2. (2 pts) ISL Book: Problem 8.2 (Page 332)
3. (2 pts) ESL Book: Problem 10.1 (Page 384)
4. (2 pts) ESL Book: Problem 10.2 (Page 384)