# STAT 154: Homework 7 solution

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.
   **Ans. False. Logistic regression puts no assumptions on how the data points are generated**

2. Logistic regression cannot be kernelized.
   **Ans. False. See lab discussions.**

3. Logistic Regression can be used for regression (when the response $y$ take continuous value in $\mathbb{R}$).
   **Ans. False. The Bernoulli distribution can not handle continuous output.**

4. Hard-margin SVM still return a classifier if the training data is not linearly separable.
   **Ans. False. The optimization problem is no longer feasible 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 $f(x) = w^\top x + b$.
   **Ans. True. In any constrained optimization, the constraints that are not satisfied are not necessary to compute the final solution.**

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).
   **Ans. False. There will be less points. In the regime, where $C$ tends to infinity, we get the hard-margin SVM, which allows no point inside the margin.**

7. As $C$ approaches 0, the soft margin SVM solution approaches that of the hard margin SVM.
   **Ans. False.**

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.
   **Ans. False. Since we have more constraints, the optimal solution will be a feasible solution in the original soft-margin SVM. As a consequence, the minimum objective value is equal or larger.**

## 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.**

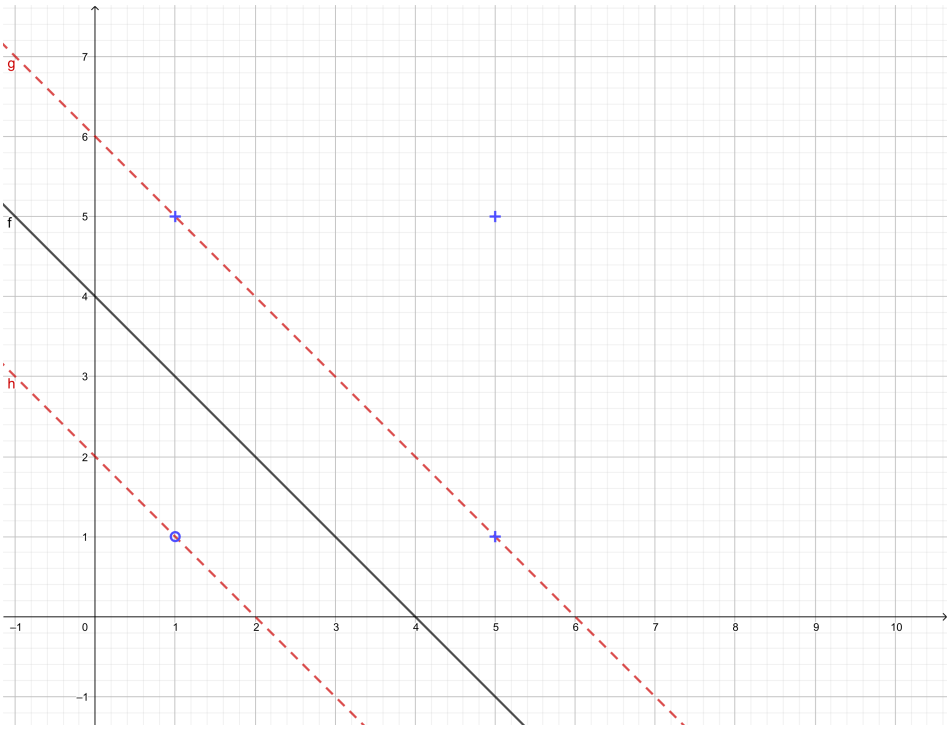


Figure 1: Figure for 5.1

2. (1 pt) By visual examination, **guess the coordinates of the weight vector $w$ (up to a constant).**
   **Ans. $(1,1)^\top$ up to constant factor (previous solution had a mistake here).**

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 $\pm 1$. For this, note

that the distance between those two margin-defining hyperplanes is equal to $d = 2/\|w\|_2$.

**Ans. By visual inspection, the distance between the two margin-defining hyperplanes is $2\sqrt{2}$. Thus we should take $\|w\|_2 = \frac{1}{\sqrt{2}}$. With the direction from previous question, we have $w = (\frac{1}{2}, \frac{1}{2})^\top$.**

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 $f(x_1) = -1$.**
**Ans. $w = (\frac{1}{2}, \frac{1}{2})^\top$. Plugging in $f(x_1) = -1$, we have**

$$\begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}^\top \begin{bmatrix} 1 \\ 1 \end{bmatrix} + b = -1.$$

**Thus $b = -2$. *And thus the equation of the solid line is***

$$x_1 + x_2 - 4 = 0.$$

***And once again, it can be computed directly from the graph.***

## 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., $g_w \in \partial f(w)$.**
**Ans. Let $k$ be an index of the function $f_k(w)$ that achieves the maximum at $w$. Since $g_w \in \partial f_k(w)$, we have**

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

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

**Since $f(z) \geq f_k(z)$ and $f(w) = f_k(w)$, we conclude that**

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

**This, by definition, means that $g_w$ is a valid subgradient of $f$ at $w$.**

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.
**Ans.** *$f$ is the maximum of two functions $f_1 : w \mapsto 0$ and $f_2 : w \mapsto 1 - yw^\top x$. Both functions are convex and differentiable, with*

$$\nabla f_1(w) = 0 \text{ and } \nabla f_2(w) = -yx.$$

**According to result from the previous question, the subgradient of $f$ will depend on which function achieves the maximum. Hence, one subgradient is**

$$g_w = \begin{cases} 0, & \text{if } 0 \geq 1 - yw^\top x. \\ -yx, & \text{otherwise.} \end{cases}$$

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

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

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

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

[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 $y_i(\widetilde{w}^\top \widetilde{x}) > 0$.

**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:** $w^{(k)}$

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.
**Ans. A classfier based on the sign of $\hat{y}$ is correct if the predicted response $\hat{y}$ to have the same sign as $y$. The Perceptron loss is zero if if $\hat{y}$ and $y$ have the same sign and is non-zero otherwise. A classifier with perceptron loss zero will also achieve zero classification error.**
**By definition, a separting hyperplance correctly classifies all training data points. This means that $\hat{y}_i$ and $y_i$ always have the same sign. Thus the perceptron loss is zero.**

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 $\alpha$ is the step-size. To derive the sub-gradient, use results from questions 2.1 and 2.2.*

**Ans. We first derive the subgradient for the perceptron loss for one data point**

$$f_i = \ell(x_i^\top w, y_i) = \max\left\{0, -yx_i^\top w\right\}.$$

**Applying the same reasoning we used in question 2.1, we obtain a subgradient of $f_i$**

$$g_{i,w} = \begin{cases} -yx_i, & \textbf{if } -yx_i^\top w \leq 0 \\ 0, & \textbf{otherwise} \end{cases}$$

**We observe that the if-else in Algorithm 1 is exactly following the two cases in the subgradient calculation. The for-loop does subgradient descent on the loss for one data point at a time. The *allCorrect* variable ensures that the algorithm terminates when all data points are correctly classified.**

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 $\alpha \in \mathbb{R}^n$.**

   **Ans. The perceptron algorithm in each step either adds $y_i x_i$ (for some $i$) to the weight $w$ or does nothing. Thus we can conclude that the final weight is of the form**

$$\hat{w} = \sum_{i=1}^n \alpha_i x_i.$$

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 $\ell_2$-regularized formulation of the SVM objective

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

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.*
**Ans. We apply the same reasoning we used in question 2.1. $J_i(w)$ is the maximum of two functions $f_1(w) = \frac{\lambda}{2} \|w\|_2^2$ and $f_2(w) = \frac{\lambda}{2} \|w\|_2^2 + 1 - y_i w^\top x_i$. Both functions are convex and differentiable,**

$$\nabla f_1(w) = \lambda w \text{ and } \nabla f_2(w) = \lambda w - y_i x_i.$$

**We conclude using the fact the subgradient of the subfunction that achieves the maximum is also a subgradient of the function $J_i$.**

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.*
**Ans. The subgradient update for $i$-th data point with step-size $\eta$ is as follows,**

$$w \leftarrow w - \eta g_{i,w}.$$

**Plugging in the subgradient from 2.6, we obtain**

$$w \leftarrow \begin{cases} w - \eta\left(\lambda w - y_i x_i\right) & \textbf{if } 1 - y_i w^\top x_i > 0 \\ w - \eta \cdot \lambda w. \end{cases}$$

**Simplying the update equation, we get exactly the updates in the Pegasos algorithm.**

## 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:

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

1. (3 pts) **Derive the expression for the logistic loss as a function of $\beta$, 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 $\nabla_\beta^2 \mathcal{L}(\beta)$.** You may use the shorthand notation $p_i = \mathbb{P}(y_i = 1 | x_i; \beta)$.

**Ans. Using the fact that data is i.i.d. generated and the conditional likelihood for one data point in equation (4), we have the negative conditional log-likelihood for all data points,**

$$\begin{aligned} \mathcal{L}(\beta) &= -\log \mathbb{P}(y_1, \ldots, y_n | x_1, \ldots, x_n; \beta) \\ &= -\sum_{i=1}^n \log \mathbb{P}(Y_i = y_i | x_i; \beta) \\ &= -\log \left( \frac{e^{y_i \beta^\top x_i}}{1 + e^{\beta^\top x_i}} \right), \quad \textbf{for } y_i \in \{0, 1\} \\ &= -y_i \beta^\top x_i + \log\left(1 + e^{\beta^\top x_i}\right). \end{aligned}$$

**The gradient is**

$$\nabla_\beta \mathcal{L}(\beta) = \sum_{i=1}^n -y_i x_i + \frac{e^{\beta^\top x_i}}{1 + e^{\beta^\top x_i}} x_i.$$

**Let $f(t) = \frac{e^t}{1+e^t} = 1 - \frac{1}{1+e^t}$. We have**

$$f'(t) = \frac{e^t}{(1 + e^t)^2}.$$

**Hence for the Hessian, we have**

$$\nabla_\beta^2 \mathcal{L}(\beta) = \sum_{i=1}^{n} \frac{e^{\beta^\top x_i}}{\left(1+e^{\beta^\top x_i}\right)^2} x_i x_i^\top = \sum_{i=1}^{n} p_i(1-p_i) x_i x_i^\top .$$

**We can see this as the weighted version of the covariance matrix.**

$$\nabla_\beta^2 \mathcal{L}(\beta) = \mathbf{X}^\top \mathbf{W} \mathbf{X},$$

**where** $\mathbf{X} = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix}$ **is the design matrix and the diagonal weight matrix** $\mathbf{W} = \operatorname{diag}\left(p_1(1-p_1), \ldots, p_n(1-p_n)\right)$**. Using this matrix notation, the gradient can be writen as**

$$\nabla_\beta \mathcal{L}(\beta) = \mathbf{X}^\top (\mathbf{p} - \mathbf{y}),$$

**where** $\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$ **and** $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$**.**

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.*
**Ans. Since** $W$ **has nonnegative diagonal values, the hessian matrix** $\nabla_\beta^2 \mathcal{L}(\beta) = \mathbf{X}^\top \mathbf{W} \mathbf{X}$ **is a positive-semidefinite matrix. Thus the loss is convex.**

3. (3 pts) Newton's method is effective for minimizing a convex function or maximizing a concave function. **Given an iterate** $\beta$**, 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 $p_i = \mathbb{P}(y_i = 1 | x_i; \beta)$.
**Ans. The Newton's method updates the weight iteratively with the equation**

$$\beta \leftarrow \beta - \eta \left(\nabla_\beta^2 \mathcal{L}(\beta)\right)^{-1} \nabla_\beta \mathcal{L}(\beta),$$

**where** $\eta$ **is the step-size. Plugging the gradient and hessian expression from previous question, we obtain the update equation.**

$$\beta \leftarrow \beta + \eta \left(\mathbf{X}^\top \mathbf{W} \mathbf{X}\right)^{-1} \mathbf{X}^\top (\mathbf{y} - \mathbf{p}),$$

4. (2 pts) **Justify why this method is called iteratively re-weighted least squares (IRWLS/IRLS) for short.**
**Ans. We define the weighted design matrix $\tilde{\mathbf{X}}$ and the weighted response $\tilde{\mathbf{y}}$ as follows,**

$$\tilde{\mathbf{X}} = \sqrt{\mathbf{W}}\mathbf{X} \text{ and } \tilde{\mathbf{y}} = \sqrt{\mathbf{W}}^{-1}(\mathbf{y} - \mathbf{p}).$$

**The Newton updates become**

$$\beta \leftarrow \beta + \eta \left(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{y}}.$$

**The expression $\left(\tilde{\mathbf{X}}^\top \tilde{\mathbf{X}}\right)^{-1} \tilde{\mathbf{X}}^\top \tilde{\mathbf{y}}$ is the solution of the least squares problem**

$$\min_\beta \left\|\tilde{\mathbf{X}}\beta - \tilde{\mathbf{y}}\right\|_2^2.$$

**Thus the Newton's method on the negative log-likelihood of the logistic regression can be seen as updating the $\beta$ iteratively with the solution of a weighted least squares problem.**

# 5 Decision Trees and Ada-boost (9 pts)

1. (3 pts) ISL Book: Problem 8.1 (Page 332)
**Ans. Like Figures 8.1 and 8.2, we have a decision tree with 6 regions.**

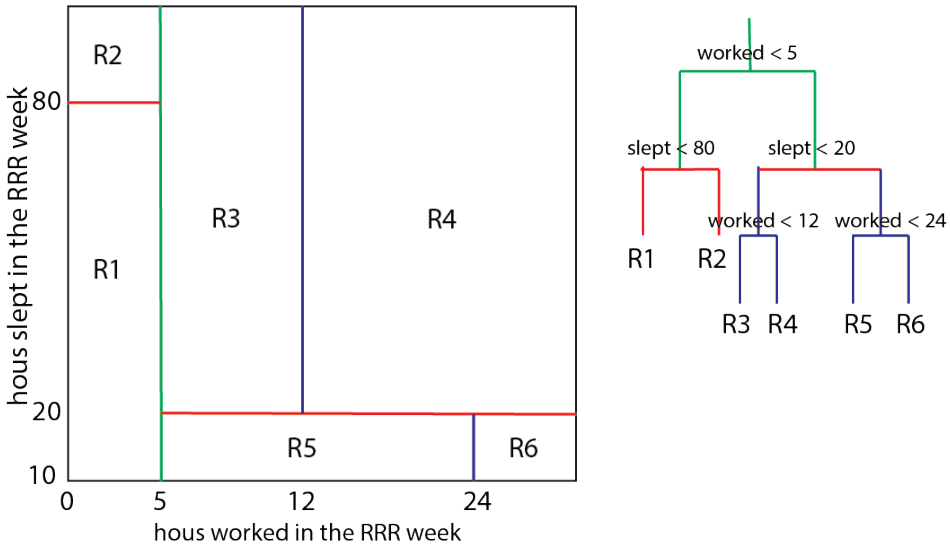


Figure 2: Figure for 5.1

2. (2 pts) ISL Book: Problem 8.2 (Page 332)
**Ans. By definition of the Boosting algorithm (Algorithm 8.2), the result**

**from Boosting is always a sum of base learners. The only thing we need to explain is that why each base learner here only depneds on one feature $X_j$. This is because we are using depth-one trees (or stumps) as base learners. The single split in the depth-one tree only depends on one feature $X_j$. Hence after combining the functions depending on the same feature, the final output can be written as the following additive model**

$$\sum_{j=1}^{p} f_j(X_j).$$

3. (2 pts) ESL Book: Problem 10.1 (Page 384)
**Ans. According to equation (10.13), the minimized weighted error rate $\mathbf{err}_m$ is defined as**

$$\mathbf{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m) I(y_i \neq G_m(x_i))}}{\sum_{i=1}^{N} w_i^{(m)}}.$$

**Taking derivative of equation (10.11) with respect to $\beta$, we obtain**

$$\left(e^{\beta} + e^{-\beta}\right) \cdot \sum_{i=1}^{N} w_i^{(m)} I(y_i \neq G(x_i)) - e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)} = 0.$$

**Dividing both sides by $e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)}$, we obtain**

$$\left(e^{2\beta} + 1\right) \mathbf{err}_m - 1 = 0.$$

**We solve for the optimal $\beta$ and obtain**

$$\beta_m = \frac{1}{2} \log \frac{1 - \mathbf{err}_m}{\mathbf{err}_m}.$$

4. (2 pts) ESL Book: Problem 10.2 (Page 384)
**Ans.**

$$\mathbb{E}_{Y|x}\left(e^{-Yf(x)}\right) = e^{-f(x)} \cdot \Pr(Y = 1 \mid x) + e^{f(x)} \cdot \Pr(Y = -1 \mid x).$$

**Since the minimization over $f$ allows any function $f$, we can simply consider the minimum of the following function**

$$\begin{aligned} g : \mathbb{R} &\to \mathbb{R} \\ t &\mapsto e^{-t} \cdot \Pr(Y = 1|x) + e^{t} \cdot \Pr(Y = -1 \mid x). \end{aligned}$$

**Taking derivative of $g$, we have**

$$g'(t) = -e^{-t} \cdot \Pr(Y = 1 \mid x) + e^{t} \cdot \Pr(Y = -1 \mid x)$$

**Observe that $g''(t)$ is nonnegative, the minimum is taken by solving $g'(t) = 0$. Hence**

$$f^*(x) = t^* = \frac{1}{2} \log \frac{\Pr(Y = 1 \mid x)}{\Pr(Y = -1 \mid x)}.$$