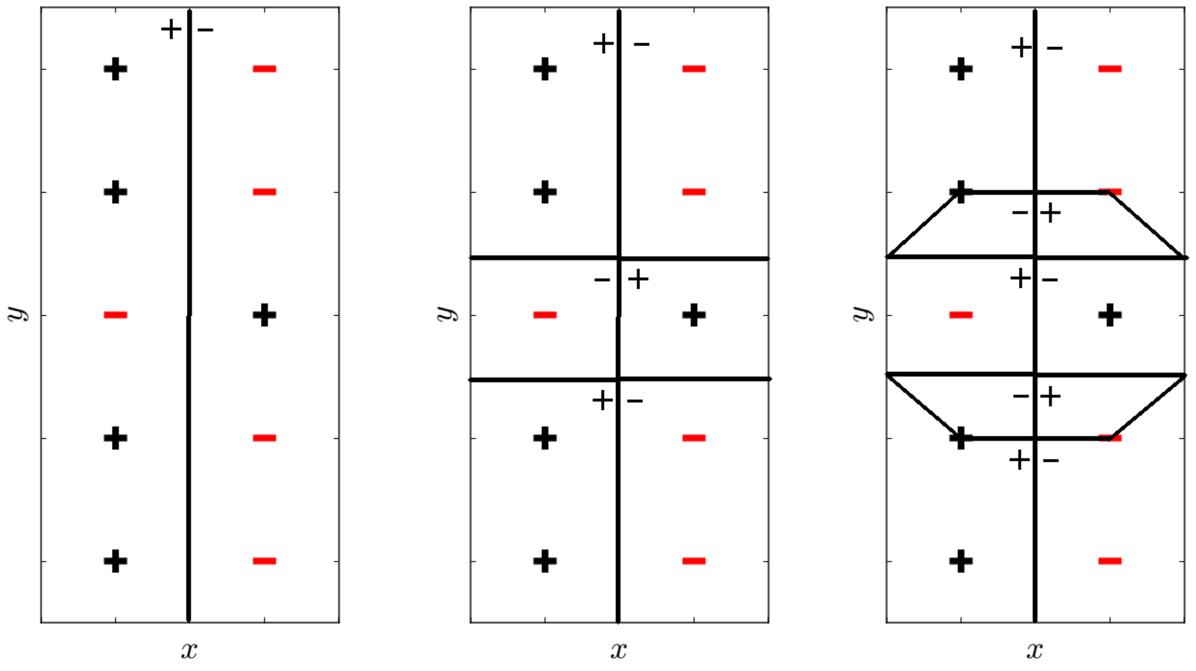
CSE 546 Midterm Exam, Fall 2014(with Solution)

1. Personal info:
   - Name:
   - UW NetID:
   - Student ID:
2. There should be 14 numbered pages in this exam (including this cover sheet).
3. You can use any material you brought: any book, class notes, your print outs of class materials that are on the class website, including my annotated slides and relevant readings. You cannot use materials brought by other students.
4. If you need more room to work out your answer to a question, use the back of the page and clearly mark on the front of the page if we are to look at what's on the back.
5. Work efficiently. Some questions are easier, some more difficult. Be sure to give yourself time to answer all of the easy ones, and avoid getting bogged down in the more difficult ones before you have answered the easier ones.
6. You have 80 minutes.
7. Good luck!

Question	Topic	Max score	Score
1	Short Answer	24	
2	Decision Trees	16	
3	Logistic Regression	20	
4	Boosting	16	
5	Elastic Net	24	
Total		100	

1 [24 points] Short Answer

1. [6 points] On the 2D dataset below, draw the decision boundaries learned by the following algorithms (using the features x/y). **Be sure to mark which regions are labeled positive or negative, and assume that ties are broken arbitrarily.**



(a) Logistic regression ($\lambda = 0$) (b) 1-NN (c) 3-NN

2. [6 points] A random variable follows an *exponential* distribution with parameter λ ($\lambda > 0$) if it has the following density:

$$p(t) = \lambda e^{-\lambda t}, \quad t \in [0, \infty)$$

This distribution is often used to model waiting times between events. Imagine you are given i.i.d. data $T = (t_1, \ldots, t_n)$ where each t_i is modeled as being drawn from an exponential distribution with parameter λ.

(a) [3 points] Compute the log-probability of T given λ. (Turn products into sums when possible).

★ **ANSWER:**

$$\begin{aligned}
\ln p(T) &= \ln \prod_i p(t_i) \\
\ln p(T) &= \sum_i \ln(\lambda e^{-\lambda t_i}) \\
\ln p(T) &= \sum_i \ln \lambda - \lambda t_i \\
\ln p(T) &= \boxed{n \ln \lambda - \lambda \sum_i t_i}
\end{aligned}$$

(b) [3 points] Solve for $\hat{\lambda}_{MLE}$.

★ **ANSWER:**

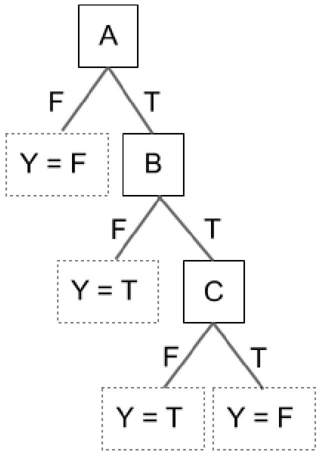
$$\begin{aligned}
\frac{\partial}{\partial \lambda}(n \ln \lambda - \lambda \sum_i t_i) &= 0 \\
\frac{n}{\lambda} - \sum_i t_i &= 0 \\
\hat{\lambda}_{MLE} &= \boxed{\frac{n}{\sum_i t_i}}
\end{aligned}$$

3. [6 points]

A	B	C	Y
F	F	F	F
T	F	T	T
T	T	F	T
T	T	T	F

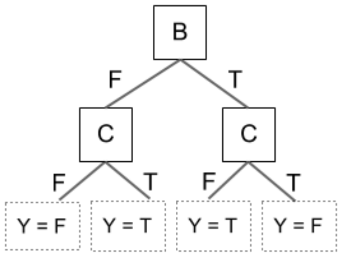
(a) [3 points] Using the dataset above, we want to build a decision tree which classifies Y as T/F given the binary variables A, B, C. Draw the tree that would be learned by the greedy algorithm with zero training error. You do not need to show any computation.

★ **ANSWER:**



(b) [3 points] Is this tree optimal (i.e. does it get zero training error with minimal depth)? Explain in less than two sentences. If it is not optimal, draw the optimal tree as well.

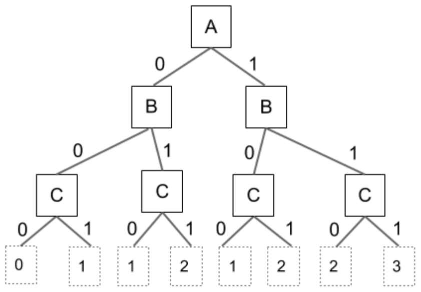
★ **ANSWER:** Although we get a better information gain by first splitting on A, Y is just a function of B/C: $Y = B \text{xor} C$. Thus, we can build a tree of depth 2 which classifies correctly, and is optimal.



4. [6 points] In class we used decision trees and ensemble methods for classification, but we can use them for regression as well (i.e. learning a function from features to real values). Let's imagine that our data has 3 binary features A, B, C, which take values 0/1, and we want to learn a function which counts the number of features which have value 1.

(a) [2 points] Draw the decision tree which represents this function. How many leaf nodes does it have?

★ **ANSWER:**



We can represent the function with a decision tree containing $\boxed{\text{8 nodes}}$.

(b) [2 points] Now represent this function as a sum of decision stumps (e.g. $sgn(A)$). How many terms do we need?

★ **ANSWER:**

$$f(x) = sgn(A) + sgn(B) + sgn(C)$$

Using a sum of decision stumps, we can represent this function using $\boxed{\text{3 terms}}$.

(c) [2 points] In the general case, imagine that we have d binary features, and we want to count the number of features with value 1. How many leaf nodes would a decision tree need to represent this function? If we used a sum of decision stumps, how many terms would be needed? No explanation is necessary.

★ **ANSWER:** Using a decision tree, we will need $\boxed{2^d \text{ nodes}}$. Using a sum of decision stumps, we will need $\boxed{d \text{ terms}}$.

2 [16 points] Decision Trees

We will use the dataset below to learn a decision tree which predicts if people pass machine learning (Yes or No), based on their previous GPA (High, Medium, or Low) and whether or not they studied.

GPA	Studied	Passed
L	F	F
L	T	T
M	F	F
M	T	T
H	F	T
H	T	T

For this problem, you can write your answers using $\log_2$, but it may be helpful to note that $\log_2 3 \approx 1.6$.

1. [4 points] What is the entropy H(Passed)?

★ **ANSWER:**

$$H(Passed) = -(\frac{2}{6}\log_2\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6})$$
$$H(Passed) = -(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3})$$
$$H(Passed) = \boxed{\log_2 3 - \frac{2}{3} \approx 0.92}$$

2. [4 points] What is the entropy H(Passed | GPA)?

★ **ANSWER:**

$$H(Passed|GPA) = -\frac{1}{3}(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) - \frac{1}{3}(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) - \frac{1}{3}(1\log_2 1)$$
$$H(Passed|GPA) = \frac{1}{3}(1) + \frac{1}{3}(1) + \frac{1}{3}(0)$$
$$H(Passed|GPA) = \boxed{\frac{2}{3} \approx 0.66}$$

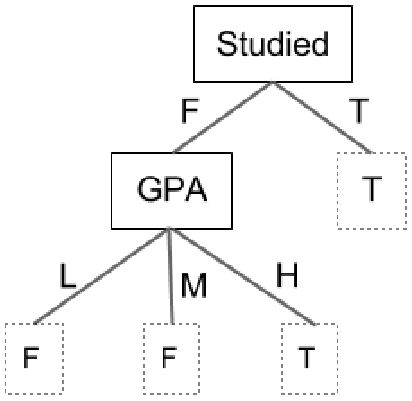
3. [4 points] What is the entropy H(Passed | Studied)?

★ **ANSWER:**

$$H(Passed|Studied) = -\frac{1}{2}(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}) - \frac{1}{2}(1\log_2 1)$$
$$H(Passed|Studied) = \frac{1}{2}(\log_2 3 - \frac{2}{3}$$
$$H(Passed|Studied) = \boxed{\frac{1}{2}\log_2 3 - \frac{1}{3} \approx 0.46}$$

4. [4 points] Draw the full decision tree that would be learned for this dataset. You do not need to show any calculations.

★ **ANSWER:** We want to split first on the variable which maximizes the information gain $H(Passed) - H(Passed|A)$. This is equivalent to minimizing $H(Passed|A)$, so we should split on "Studied?" first.



3 [20 points] Logistic Regression for Sparse Data

In many real-world scenarios our data has millions of dimensions, but a given example has only hundreds of non-zero features. For example, in document analysis with word counts for features, our dictionary may have millions of words, but a given document has only hundreds of unique words. In this question we will make l_2 regularized SGD efficient when our input data is sparse. Recall that in l_2 regularized logistic regression, we want to maximize the following objective (**in this problem we have excluded w_0 for simplicity**):

$$F(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2}\sum_{i=1}^{d} \mathbf{w}_i^2$$

where $l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w})$ is the logistic objective function

$$l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) = y^{(j)}(\sum_{i=1}^{d} \mathbf{w}_i \mathbf{x}_i^{(j)}) - \ln(1 + \exp(\sum_{i=1}^{d} \mathbf{w}_i \mathbf{x}_i^{(j)}))$$

and the remaining sum is our regularization penalty.

When we do stochastic gradient descent on point $(\mathbf{x}^{(j)}, y^{(j)})$, we are approximating the objective function as

$$F(\mathbf{w}) \approx l(\mathbf{x}^{(j)}, y^{(j)}, \mathbf{w}) - \frac{\lambda}{2}\sum_{i=1}^{d} \mathbf{w}_i^2$$

Definition of sparsity: Assume that our input data has d features, i.e. $\mathbf{x}^{(j)} \in \mathbb{R}^d$. In this problem, we will consider the scenario where $\mathbf{x}^{(j)}$ is sparse. Formally, let s be average number of nonzero elements in each example. We say the data is sparse when $s << d$. In the following questions, **your answer should take the sparsity of $\mathbf{x}^{(j)}$ into consideration when possible**. **Note:** When we use a sparse data structure, we can iterate over the non-zero elements in $O(s)$ time, whereas a dense data structure requires $O(d)$ time.

1. [2 points] Let us first consider the case when $\lambda = 0$. Write down the SGD update rule for $\mathbf{w}_i$ when $\lambda = 0$, using step size η, given the example $(\mathbf{x}^{(j)}, y^{(j)})$.

★ **ANSWER:** The update rule can be written as

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} + \eta \mathbf{x}_i^{(j)} \left(y^{(j)} - \frac{1}{1 + \exp(-\sum_k \mathbf{w}_k x_k^{(j)})} \right)$$

2. [4 points] If we use a dense data structure, what is the average time complexity to update $\mathbf{w}_i$ when $\lambda = 0$? What if we use a sparse data structure? Justify your answer in one or two sentences.

★ **ANSWER:** The time complexity to calculate $\sum_k \mathbf{w}_k x_k^{(j)}$ is $O(d)$ when the data structure is dense, and $O(s)$ when the data structure is sparse. Note that even if we update $\mathbf{w}_i$ for all i, we only need to calculate $\sum_k \mathbf{w}_k x_k^{(j)}$ once, and then update the $\mathbf{w}_i$ such that $\mathbf{x}_i^{(j)} \neq 0$. So the answer is $\boxed{O(d)}$ for the dense case, and $\boxed{O(s)}$ for the sparse case.

3. [2 points] Now let us consider the general case when $\lambda > 0$. Write down the SGD update rule for $\mathbf{w}_i$ when $\lambda > 0$, using step size η, given the example $(\mathbf{x}^{(j)}, y^{(j)})$.

★ **ANSWER:**

$$\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} - \eta\lambda\mathbf{w}_i^{(t)} + \eta\mathbf{x}_i^{(j)}\left(y^{(j)} - \frac{1}{1+\exp(-\sum_k \mathbf{w}_k x_k^{(j)})}\right)$$

4. [2 points] If we use a dense data structure, what is the average time complexity to update $\mathbf{w}_i$ when $\lambda > 0$?

★ **ANSWER:** The time complexity is $O(d)$

5. [4 points] Let $\mathbf{w}_i^{(t)}$ be the weight vector after t-th update. Now imagine that we perform k SGD updates on $\mathbf{w}$ using examples $(\mathbf{x}^{(t+1)}, y^{(t+1)}), \cdots, (\mathbf{x}^{(t+k)}, y^{(t+k)})$, where $\mathbf{x}_i^{(j)} = 0$ for every example in the sequence. (i.e. the i-th feature is zero for all of the examples in the sequence). Express the new weight, $\mathbf{w}_i^{(t+k)}$ in terms of $\mathbf{w}_i^{(t)}$, k, η, and λ.

★ **ANSWER:** When $\mathbf{x}_i^{(j)} = 0$,

$$\mathbf{w}_i^{(t+1)} = \mathbf{w}_i^{(t)} - \eta\lambda\mathbf{w}_i^{(t)} = \mathbf{w}_i^{(t)}(1-\eta\lambda)$$

so the answer is

$$\mathbf{w}_i^{(t+k)} = \mathbf{w}_i^{(t)}(1-\eta\lambda)^k$$

6. [6 points] Using your answer in the previous part, come up with an efficient algorithm for regularized SGD when we use a sparse data structure. What is the average time complexity per example? (Hint: when do you need to update $\mathbf{w}_i$?)

Algorithm 1: Sparse SGD Algorithm for Logistic Regression with Regularization

```
Initialize $c_i \leftarrow 0$ for $i \in \{1, 2, \cdots, d\}$
for $j \in \{1, 2, \cdots n\}$ do
    $\hat{p} \leftarrow \frac{1}{1+\exp(-\sum_k \mathbf{w}_k x_k^{(j)})}$
    for $i$ such that $\mathbf{x}_i^{(j)} \neq 0$ do
        $k \leftarrow j - c_i$;   auxiliary variable $c_i$ holds the index of last time we see $\mathbf{x}_i^{(j)} \neq 0$
        $\mathbf{w}_i \leftarrow \mathbf{w}_i(1 - \eta\lambda)^k$;   apply all the regularization updates
        $\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta \mathbf{x}_i^{(j)} \left(y^{(j)} - \hat{p}\right)$;   regularization is done in previous step
        $c_i \leftarrow j$;   remember last time we see $\mathbf{x}_i^{(j)} \neq 0$
    end
end
```

★ **ANSWER:** The idea is to only update $\mathbf{w}_i$ when $\mathbf{x}_i^{(j)} \neq 0$. Before we do the update, we apply all the regularization updates we skipped before, using the answer from previous question. You can checkout Algorithm 1 for details. Using this trick, each update takes $\boxed{O(s)}$ time. (Note: we can use the same trick applies for SGD with l_1 regularization)

4 [16 points] Boosting

Recall that Adaboost learns a classifier H using a weighted sum of weak learners h_t as follows

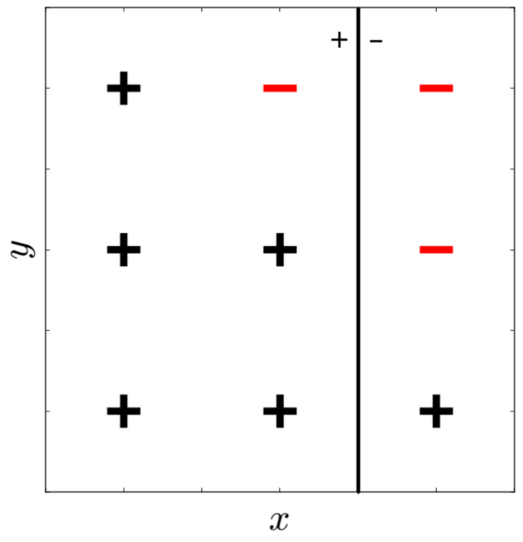
$$H(x) = sgn\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

In this question we will use decision trees as our weak learners, which classify a point as $\{1, -1\}$ based on a sequence of threshold splits on its features (here x, y).

In the questions below, be sure to mark which regions are marked positive/negative, and assume that ties are broken arbitrarily.

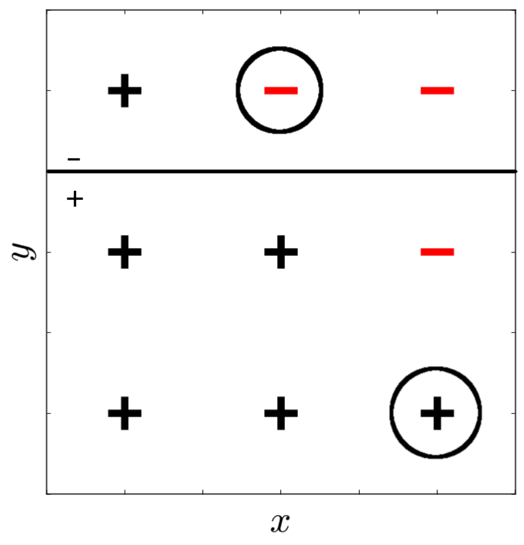
★ **COMMENT:** The solutions below are one of several possible answers. You got full credit as long as you were consistent and found a boundary with the lowest training error for h_1, h_2.

1. [2 points] Assume that our weak learners are decision trees of depth 1 (i.e. decision stumps), which minimize the weighted training error. Using the dataset below, draw the decision boundary learned by h_1.



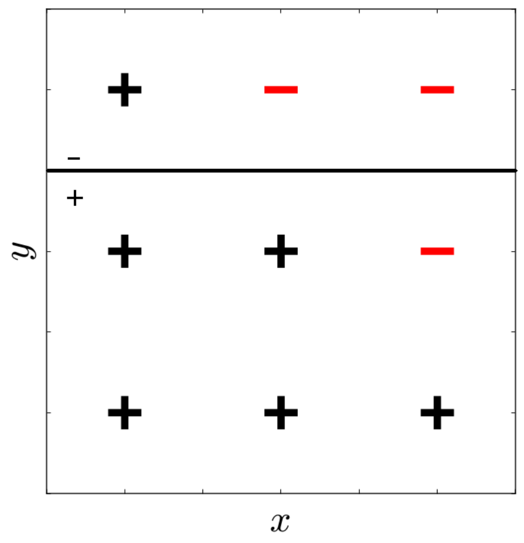
2. [3 points] On the dataset below, circle the point(s) with the highest weights on the second iteration, and draw the decision boundary learned by h_2.

★ **ANSWER:** The points with the highest weight will be those that were classified incorrectly by h_1.

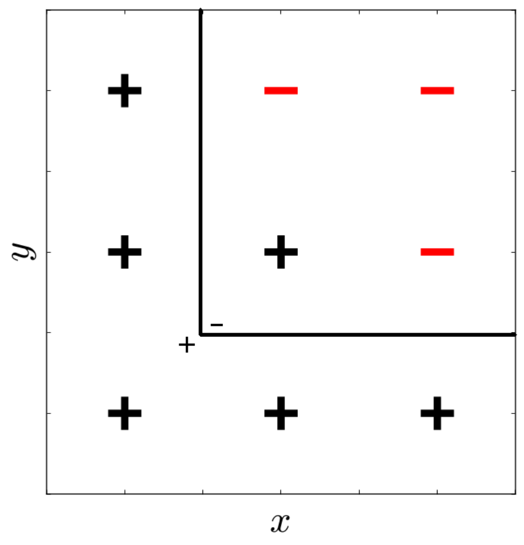


3. [3 points] On the dataset below, draw the decision boundary of $H = sgn\,(\alpha_1 h_1 + \alpha_2 h_2)$. (Hint, you do not need to explicitly compute the α's).

★ **ANSWER:** Although h_1 and h_2 misclassify the same number of points, the points which h_2 gets wrong have been downweighted. Thus, we have $\epsilon_1 > \epsilon_2$, so $\alpha_2 > \alpha_1$, and h_2 will dominate over h_1 whenever there is a conflict. In other words, H has the same boundary as h_2.

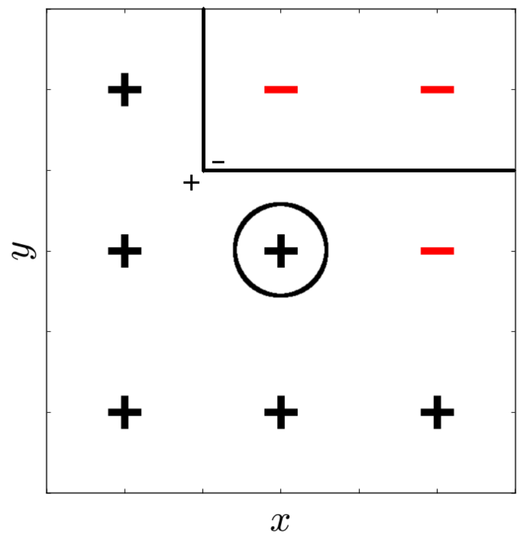


4. [2 points] Now assume that our weak learners are decision trees of maximium depth 2, which minimize the weighted training error. Using the dataset below, draw the decision boundary learned by h_1.



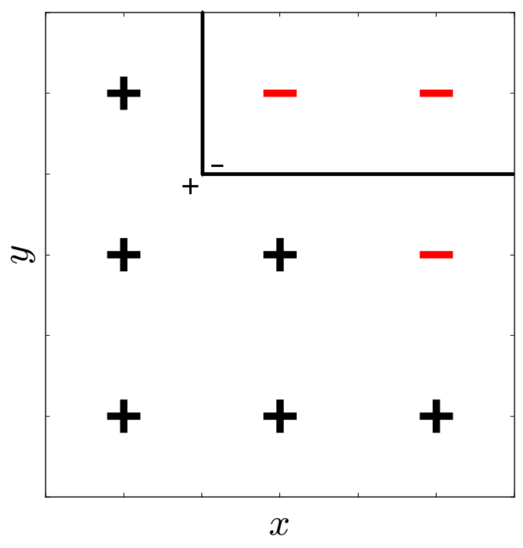
5. [3 points] On the dataset below, circle the point(s) with the highest weights on the second iteration, and draw the decision boundary learned by h_2.

★ **ANSWER:** Again, the points with the highest weight will be those that were classified incorrectly by h_1.



6. [3 points] On the dataset below, draw the decision boundary of $H = sgn\left(\alpha_1 h_1 + \alpha_2 h_2\right)$. (Hint, you do not need to explicitly compute the α's).

★ **ANSWER:** Again, we have $\alpha_2 > \alpha_1$, so h_2 will dominate over h_1 whenever there's a conflict.



5 [24 points] Elastic Net

In class we discussed two types of regularization, l_1 and l_2. Both are useful, and sometimes it is helpful combine them, giving the objective function below (**in this problem we have excluded w_0 for simplicity**):

$$F(\mathbf{w}) = \frac{1}{2}\sum_{j=1}^{n}(y^{(j)} - \sum_{i=1}^{d}\mathbf{w}_i\mathbf{x}_i^{(j)})^2 + \alpha\sum_{i=1}^{d}|\mathbf{w}_i| + \frac{\lambda}{2}\sum_{i=1}^{d}\mathbf{w}_i^2 \tag{1}$$

Here $(\mathbf{x}^{(j)}, y^{(j)})$ is j-th example in the training data, $\mathbf{w}$ is a d dimensional weight vector, λ is a regularization parameter for the l_2 norm of $\mathbf{w}$, and α is a regularization parameter for the l_1 norm of $\mathbf{w}$. This approach is called the Elastic Net, and you can see that it is a generalization of Ridge and Lasso regression: It reverts to Lasso when $\lambda = 0$, and it reverts to Ridge when $\alpha = 0$. In this question, we are going to derive the coordinate descent (CD) update rule for this objective.

Let g, h, c be real constants, and consider the function of x

$$f_1(x) = c + gx + \frac{1}{2}hx^2 \quad (h > 0) \tag{2}$$

1. [2 points] What is the x^* that minimizes $f_1(x)$? (i.e. calculate $x^* = \arg\min f_1(x)$)

★ **ANSWER:** Take the gradient of $f_1(x)$ and set it to 0.

$$g + hx = 0$$
$$x^* = \boxed{-\frac{g}{h}}$$

Let α be an additional real constant, and consider another function of x

$$f_2(x) = c + gx + \frac{1}{2}hx^2 + \alpha|x| \ (h > 0, \alpha > 0) \tag{3}$$

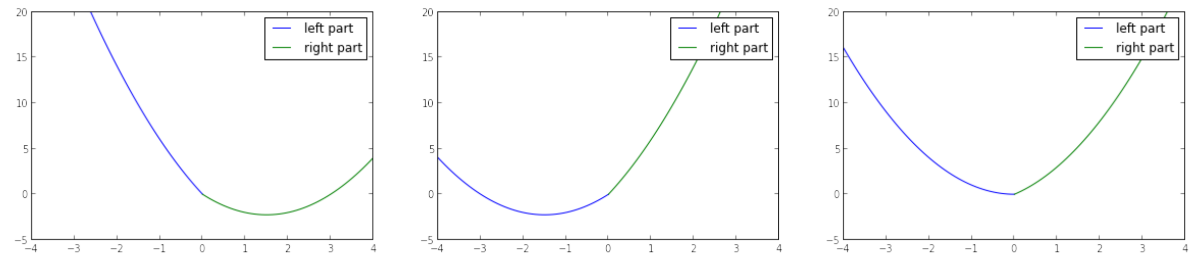
This is a piecewise function, composed of two quadratic functions:

$$f_2^-(x) = c + gx + \frac{1}{2}hx^2 - \alpha x$$

and

$$f_2^+(x) = c + gx + \frac{1}{2}hx^2 + \alpha x$$

Let $\tilde{x}^- = \arg\min_{x\in\mathbb{R}} f_2^-(x)$ and $\tilde{x}^+ = \arg\min_{x\in\mathbb{R}} f_2^+(x)$.
(**Note:** The argmin is taken over $(-\infty, +\infty)$ here)



(a) $\tilde{x}^- > 0, \tilde{x}^+ > 0$, the minima is at $\tilde{x}^+$ (b) $\tilde{x}^- < 0, \tilde{x}^+ < 0$, the minima is at $\tilde{x}^-$ (c) $\tilde{x}^- > 0, \tilde{x}^+ < 0$, the minima is at 0

Figure 1: Example picture of three cases

2. [3 points] What are $\tilde{x}^+$ and $\tilde{x}^-$? Show that $\tilde{x}^- \geq \tilde{x}^+$.

★ **ANSWER:** Using the answer from part 1), we get $\tilde{x}^+ = -\frac{g+\alpha}{h}$ and $\tilde{x}^- = -\frac{g-\alpha}{h}$. Since $\tilde{x}^- - \tilde{x}^+ = \frac{2\alpha}{h} \geq 0$, we have $\tilde{x}^- \geq \tilde{x}^-$.

3. [6 points] Draw a picture of $f_2(x)$ in each of the three cases below.
   (a) $\tilde{x}^- > 0, \tilde{x}^+ > 0$
   (b) $\tilde{x}^- < 0, \tilde{x}^+ < 0$
   (c) $\tilde{x}^- > 0, \tilde{x}^+ < 0$.

   For each case, mark the minimum as either 0, $\tilde{x}^-$, or $\tilde{x}^+$. (You do not need to draw perfect curves, just get the rough shape and the relative locations of the minima to the x-axis)

★ **ANSWER:** An example of the curves is shown in Figure 1. To understand the answer, note the following:

- $f_2^+(0) = f_2^-(0)$, this means the two piece of f_2 match each other at $x = 0$.
- When $\tilde{x}^- \geq 0$, the minimum of left part is at 0, i.e. $\arg\min_{x\in(-\infty,0]} f_2^-(x) = 0$
- When $\tilde{x}^- \leq 0$, the minimum of left part is at $\tilde{x}^-$, i.e. $\arg\min_{x\in(-\infty,0]} f_2^-(x) = \tilde{x}^-$
- Similar rules holds for the right part of the curve.

4. [4 points] Write x^*, the minimum of $f_2(x)$, as a piecewise function of g. (Hint: what is g in the cases above?)

Algorithm 2: Coordinate Descent for Elastic Net

```
while not converged do
    for $k \in \{1, 2, \cdots d\}$ do
        $g = \sum_{j=1}^{n} x_k^{(j)} (\sum_{i \neq k} \mathbf{w}_i \mathbf{x}_i^{(j)} - y_i)$
        $h = \lambda + \sum_{j=1}^{n} (\mathbf{x}_k^{(j)})^2$
        $w_k = \begin{cases} -\frac{g+\alpha}{h} & g < -\alpha \\ 0 & g \in [-\alpha, \alpha] \\ -\frac{g-\alpha}{h} & g > \alpha \end{cases}$
    end
end
```

★ **ANSWER:** Summarizing the result from the previous question, we have

$$x^* = \begin{cases} \tilde{x}^+ & \text{if} \quad \tilde{x}^+ > 0 \\ \tilde{x}^- & \text{if} \quad \tilde{x}^- < 0 \\ 0 & \text{if} \quad \tilde{x}^+ < 0, \tilde{x}^- > 0 \end{cases}$$

This is equivalent to the soft-threshold function

$$x^* = \begin{cases} -\frac{g+\alpha}{h} & \text{if} \quad g < -\alpha \\ 0 & \text{if} \quad g \in [-\alpha, \alpha] \\ -\frac{g-\alpha}{h} & \text{if} \quad g > \alpha \end{cases}$$

5. [4 points] Now let's derive the update rule for $\mathbf{w}_k$. Fixing all parameters but $\mathbf{w}_k$, express the objective function in the form of Eq (3) (here $\mathbf{w}_k$ is our x). You do not need to explicity write constant factors, you can put them all together as c. What are g and h?

$$g = \sum_{j=1}^{n} x_k^{(j)} (\sum_{i \neq k} \mathbf{w}_i \mathbf{x}_i^{(j)} - y_i) \tag{4}$$

$$h = \lambda + \sum_{j=1}^{n} (\mathbf{x}_k^{(j)})^2 \tag{5}$$

6. [5 points] Putting it all together, write down the coordinate descent algorithm for the Elastic Net (**again excluding** w_0). You can write the update for $\mathbf{w}_k$ in terms of the g and h you calculated in the previous question.

★ **ANSWER:** The algorithm is shown in Algorithm 2. You can compare it to the CD algorithm for Lasso, the only difference is adding λ to h.