Cmpsc 448 HW3

1) we start with data set $S = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots (\vec{x}_n, y_n)\}$

where $\vec{x}_i \in \mathbb{R}^d$ and labels $y_i \in \{-1, +1\}$

Since data is not linearly seperable, map to higher dimension to make data seperable.

we will add e_i to the data where e_i is a 0 vector except for the i-th position which will be 1.

we want to find weight vector $\vec{w} \in \mathbb{R}^{d+n}$

$y_i(w^T \vec{x}_i) > 0 \quad \forall i \in \{1 \ldots n\}$

$\vec{w} = [w_x^T, w_e^T]^T$ since data is not seperable let $w_x^T = 0$

$w_e = [y_1, y_2 \ldots y_n]^T \in \mathbb{R}^n$

$w^T \vec{x}_i = w_e^T e_i$ since e_i has 1 at i and 0s elsewhere $w^T \vec{x}_i = y_i$

therefore $y_i(w^T \vec{x}_i) = y_i^2$ since $y_i \in \{-1, +1\}$ $y_i^2 = 1$ which is > 0 always

2) $f(w) = \sum \log(1 + e^{-y_i w^T x_i})$

a. $\nabla f(w) = \sum \frac{-y_i x_i e^{-y_i w^T x_i}}{1 + e^{-y_i w^T x_i}} = \sum \frac{-y_i x_i}{(1 + e^{-y_i w^T x_i})(e^{y_i w^T x_i})} = \sum \frac{-y_i x_i}{1 + e^{y_i w^T x_i}}$

b. Psudo code:

```
Function(data = {(x1,y1) ... (xn,yn)}, learnrate, iterations)
    for i = 0 to iterations
        calculate ∇f(w)
        gradient = 0
        for every data point in data
            prob = Sigmoid(xi, yi)
            gradient = gradient + (prob - 1) * yi * xi
        w = w - learnrate * gradient
    return w
```

c. if the data is linearly seperable, the optimization problem has no finite solution because logistic regression models use the sigmoid function for probabilities

If there exists a perfect linear seperator, $y_i w^T x_i \to 0$

As gradient descent works, the weights will keep increasing in order to make $y_i w^T x_i$ either 0 or 1 probability

This makes $w^T x_i \to \infty$ for correctly classified points.

Since the sigmoid function approaches 1 as $z \to \infty$, the loss function keeps decreasing. The weights will keep getting larger as Gradient descent keeps running, leading to some of the weights being ∞

3) $P[y=k|x;w_1 \dots w_n] = \frac{e^{w_k^T x}}{\sum_{i=1}^{K} e^{w_i^T x}}$

$\sum P(y=k|x;w_1 \dots w_n] = 1 \qquad w_K = 0 \qquad f^{(multi)}(x) = \underset{k \in \{1 \dots K\}}{\operatorname{argmax}} P[y=k|x]$

assume each y_n is indpendnt so $L(\theta) = \prod_{i=1}^{n} P[y_i|x_i;w_1 \dots w_K]$

a) $\log(P(y_1 \dots y_n|x_1 \dots x_n;w_1 \dots w_n)) = \sum \log P(y_i|x_i;w_1 \dots w_n)$

$$l(\theta) = \sum_{i=1}^{n} \log\left(\frac{e^{w_{y_i}^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}}\right) = \sum_{i=1}^{n}\left[w_{y_i}^T x_i - \log \sum_{j=1}^{K} e^{w_j^T x_i}\right]$$

since we want to minimize the loss, we minimize the negative log likelihood

$$l(w) = -\sum_{i=1}^{n}\left[w_{y_i}^T x_i - \log \sum_{j=1}^{K} e^{w_j^T x_i}\right]$$

b) gradient of n likelihood with respect to w_K

$$\frac{\partial l(w)}{\partial w_K} = \sum_{i=1}^{n}\left[I\{y_i=K\} x_i - \frac{e^{w_K^T x_i}}{\sum_{j=1}^{K} e^{w_j^T x_i}} x_i\right]$$

$$= \sum_{i=1}^{n} x_i \left[I_{\{y_i=k\}} - P(y=k|x_i,w)\right]$$

c) stochastic gradient descent update

$$w_k = w_k + \eta x_i \left[I_{\{y_i=k\}} - P(y=k|x_i,w)\right]$$

4) $X = \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ -2 & 0 \end{bmatrix} \quad y = \begin{bmatrix} -1 \\ -1 \\ +1 \end{bmatrix}$

$\min_{w,b} \frac{1}{2}\|w\|_2^2$ such that $y_i(w^T x_i + b) \geq 1, \; i = 1, 2 \dots n$

$\vec{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

Point 1) $-1(w_1(0) + w_2(0) + b) \geq 1 \qquad -b \geq 1$

Point 2) $-1(w_1(0) + w_2(-1) + b) \geq 1 \qquad w_2 - b \geq 1$

Point 3) $+1(w_1(-2) + w_2(0) + b) \geq 1 \qquad -2w_1 + b \geq 1$

$b = -1 \rightarrow \quad w_2 + 1 \geq 1 \qquad -2w_1 - 1 \geq 1$

$w_2 = 0 \qquad -2w_1 \geq 2$

$w_1 \leq -1$

$w^* = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad b = -1$

5) data $= (a,1), (-a,-1) \qquad a > 0 \qquad n = 2 \quad d = 1$

Hard SVM: $\min_{w,b} \frac{1}{2} w^2$ st $y_i(wx_i + b) \geq 1$

$1(w(a) + b) \geq 1 \qquad -1(w(-a) + b) \geq 1 \rightarrow \quad wa + b \geq 1 \qquad wa - b \geq 1$

$b = 0$, $w = \frac{1}{a}$

Soft SVM: $\min_{w,b,\xi_i} \frac{1}{2} w^2 + C \sum_{i=1}^{n} \xi_i$ st $y_i(wx_i + b) \geq 1 - \xi_i \qquad \xi_i \geq 0$

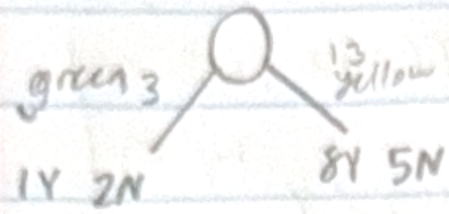
$wa + b \geq 1 - \xi_1 \qquad wa - b \geq 1 - \xi_2$

for large C, $\xi_i = 0$ for $i = 1, 2$ which will result in same solution $b = 0$, $w = \frac{1}{a}$

6 decision trees

Parent node: 7/16 no $-\frac{7}{16}\log_2\left(\frac{7}{16}\right) - \frac{9}{16}\log_2\left(\frac{9}{16}\right) = 0.9887$

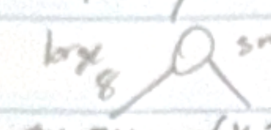
split by color: $P(Y|\text{green}) = -\frac{1}{3}\log_2\left(\frac{1}{3}\right) - \frac{2}{3}\log_2\left(\frac{2}{3}\right) = 0.9183$



$P(Y|\text{yellow}) = -\frac{8}{13}\log_2\left(\frac{8}{13}\right) - \frac{5}{13}\log_2\left(\frac{5}{13}\right) = 0.9612$

$\text{Infogain (color)} = H(Y) - \left[\frac{3}{16}H(Y|\text{green}) + \frac{13}{16}H(Y|\text{yellow})\right] = 0.0355$

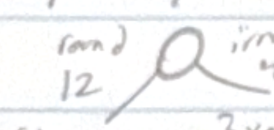
split by size: $P(Y|\text{Large}) = -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right) = 0.9544$



$P(Y|\text{Small}) = -\frac{6}{8}\log_2\left(\frac{6}{8}\right) - \frac{2}{8}\log_2\left(\frac{2}{8}\right) = 0.8113$

$\text{Infogain(size)} = H(Y) - \left[\frac{1}{2}H(Y|\text{Large}) + \frac{1}{2}H(Y|\text{Small})\right] = 0.1059$

split by shape: $P(Y|\text{round}) = -\frac{6}{12}\log_2\left(\frac{6}{12}\right) - \frac{6}{12}\log_2\left(\frac{6}{12}\right) = 1$



$P(Y|\text{irregular}) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) = 0.8113$

$\text{Infogain (shape)} = H(Y) - \left[\frac{12}{16}H(Y|\text{round}) + \frac{4}{16}H(Y|\text{irregular})\right] = 0.0359$

Split by size because it results in the highest info gained.

Cmpsc 448 HW3 Question 7

First, we have section 7.1, which is titled the optimization framework for liner models. In this section, we are concerned with optimizing various linear models. The text specifically mentions a classification algorithm, named perceptron, which is an algorithm that we covered in class. The goal is to create a hyperplane to separate the data into two different classes. But the problem with this is that not all data is linearly separable, which will cause the perceptron algorithm to run infinitely. So the goal is to find other linear models that can try to separate the data, and in the case that the data is not separable, we want the model that makes the fewest errors. So, this optimization problem is that we want to minimize the number of errors that the model makes. We want to try and find a way to optimize the 0/1 loss function, which will reduce the training error. We also want to make sure that we don't overfit the data, so we introduce a regularization term to the loss function.

Section 7.2 Convex Surrogate Loss Functions. Like what we saw previously, the 0/1 loss function is not differentiable, which means it is very difficult to optimize. We know that convex functions are easy to minimize through calculus. Using this information, we can replace the 0/1 loss function with a convex surrogate loss function, which is similar to 0/1 loss in shape.

Section 7.3 Weight Regularization, is about trying to fund a good regularization term to make sure that the model does not overfit the data. The text discusses two possible regularization terms: the sum of the squared weights or use the sum of absolute weights. Both of these terms are convex, so the loss function will stay easy to optimize.

In section 7.4, optimization using gradient descent is discussed. An analogy is used comparing gradient descent as trying to get down a mountain while blindfolded. If we keep taking small steps downwards, we will eventually reach the bottom. This is the idea behind gradient descent, which can be implemented using coding. In the code, we will first set the weights to 0, and then update them with every step down we take.

In section 7.5, we learn that we have to use subgradient optimization to try and find a gradient for where the function has multiple derivatives. The subgradient will be the set of all tangent lines to the point that is not differentiable. One example is the hinge loss function which is used for SVMs.

In section 7.6, it discusses optimizing the squared loss function. We learn about the closed form solution to optimizing this loss function. We also derive insights about when we want to use gradient descent or the closed form solution. For number of features <= 100, we use closed form and for features >= 10000, we use gradient descent.

The final section is about SVMs. We learn that the hinge loss functions help optimization in classification tasks. SVMs aim to find the hyperplane that maximizes the margin. Hard-margin SVMs require perfectly separable data, and soft-margin SVMs introduce slack variables non separable data. The margin for SVMs are inversely proportional to the norm of the weight vector, so we can use L2 regularization. C controls the tradeoff between large margins and misclassification tolerance.