CIS 472/572, Winter 2017
Written Homework 3: Logistic Regression, Neural Networks, and SVMs

Friday, February 24th 2017

Jacob Lambert, jlambert@cs.uoregon.edu

1. (a) Since we have 1-dimensional data, the positive and negative classes can be separated by the set of points for which $wx + b = 0$. So for these points, we have $x = -b/w$. The points that separate the classes fall in the range $0.5 < x < 1$. So we can classify the set of values for w and b as follows:

$$\{w, b \mid 0.5 < -b/w < 1\}$$

(b) We can easily observe that the maximum margin separator will lie on $x = 0.75$ This would give us:

$$-b/w = 0.75$$

We can also take the two nearest points and calculate the values of w, b, and $x = -b/w$.

$$w(1) + b = 1$$
$$w(0.5) + b = -1$$

Subtracting the bottom equation from the top, we get.

$$w(0.5) = 2$$

Thus

$$w = 4,\ b = -3,\ -b/w = \frac{-(-3)}{4} = 0.75$$

(c) To calculate the soft-margin SVM, we need to minimize the following:

$$\frac{w^2}{2} + C \sum_{(x,y) \in D} max(0, 1 - y(wx + b))$$

The w and b calculations were done by hand, and the svm value calculations were done using the following short python script:

```
x = [-2, -1, 0.5, 1, 2]
y = [-1, -1, -1, 1, 1]

C = 0.05

# case i omitted

# ii remove x3

w = 1.0
b = 0.0

svm = sum( max(0, 1 - yi * (w * xi + b)) for xi, yi in zip(x, y) )
svm = (w*w) / 2 + C * svm

print "ii:", svm
print "-b/w = ", -b/w

# repeat for iii - vi, and all values of C
```

i. Removing nothing
$b = -3$, $w = 4$, $-b/w = 0.75$
$C = 5$, svm value $= 8.0$
$C = 0.5$, svm value $= 8.0$
$C = 0.05$, svm value $= 8.0$

ii. Removing $\{x_3\}$
$b = 0$, $w = 1$, $-b/w = 0$
$C = 5$, svm value $= 8.0$
$C = 0.5$, svm value $= 1.25$
$C = 0.05$, svm value $= 0.575$

iii. Removing $\{x_4\}$ $b = -1.67$, $w = 1.33$, $-b/w = 1.26$
$C = 5$, svm value $= 7.63$
$C = 0.5$, svm value $= 1.56$
$C = 0.05$, svm value $= 0.952$

iv. Removing $\{x_2, x_3\}$
$b = 0.33$, $w = 0.67$, $-b/w = 0.50$
$C = 5$, svm value $= 11.85$
$C = 0.5$, svm value $= 1.39$
$C = 0.05$, svm value $= 0.341$

v. Removing $\{x_3, x_4\}$
$b = -0.33$, $w = 0.67$, $-b/w = -0.50$
$C = 5$, svm value $= 8.50$
$C = 0.5$, svm value $= 1.06$
$C = 0.05$, svm value $= 0.308$

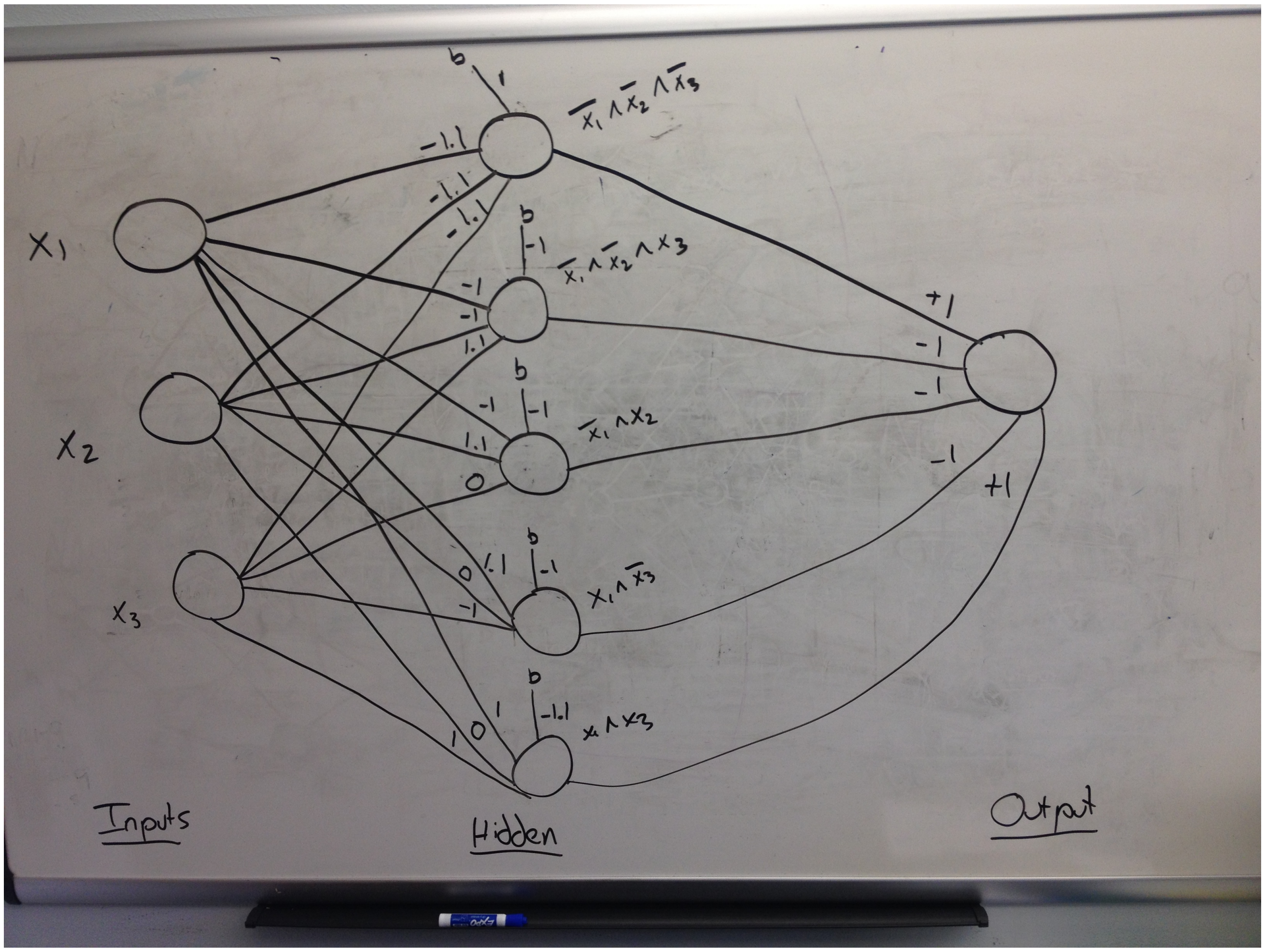
vi. Removing $\{x_2, x_3, x_4\}$
$b = 0$, $w = 0.5$, $-b/w = 0$
$C = 5$, svm value $= 11.375$
$C = 0.5$, svm value $= 1.25$
$C = 0.05$, svm value $= 0.238$

We can see that for $C = 5$, removing x_4 (case ii) gives the minimum. For $C = 0.5$, removing x_3 and x_4 gives the minimum (case v), and for $C = 0.05$, removing x_2, x_3, and x_4 gives the minimum (case vi).

2. (a) The network structure below was created by constructing individual neurons for each leaf in the decision tree. These neurons were then linked together such that only one neuron will activate for any given set of inputs, and the activation of that neuron provides the correct output. The activation function is as follows:

$$\alpha = \max(0, \text{sign}(wx + b))$$

This guarantees that all neurons where $wx + b < 0$ will give zero contribution to the output, and only the single neuron with $wx + b > 0$ will have a non-zero activation. The weights are assigned such that $wx + b \neq 0 \ \forall x$.

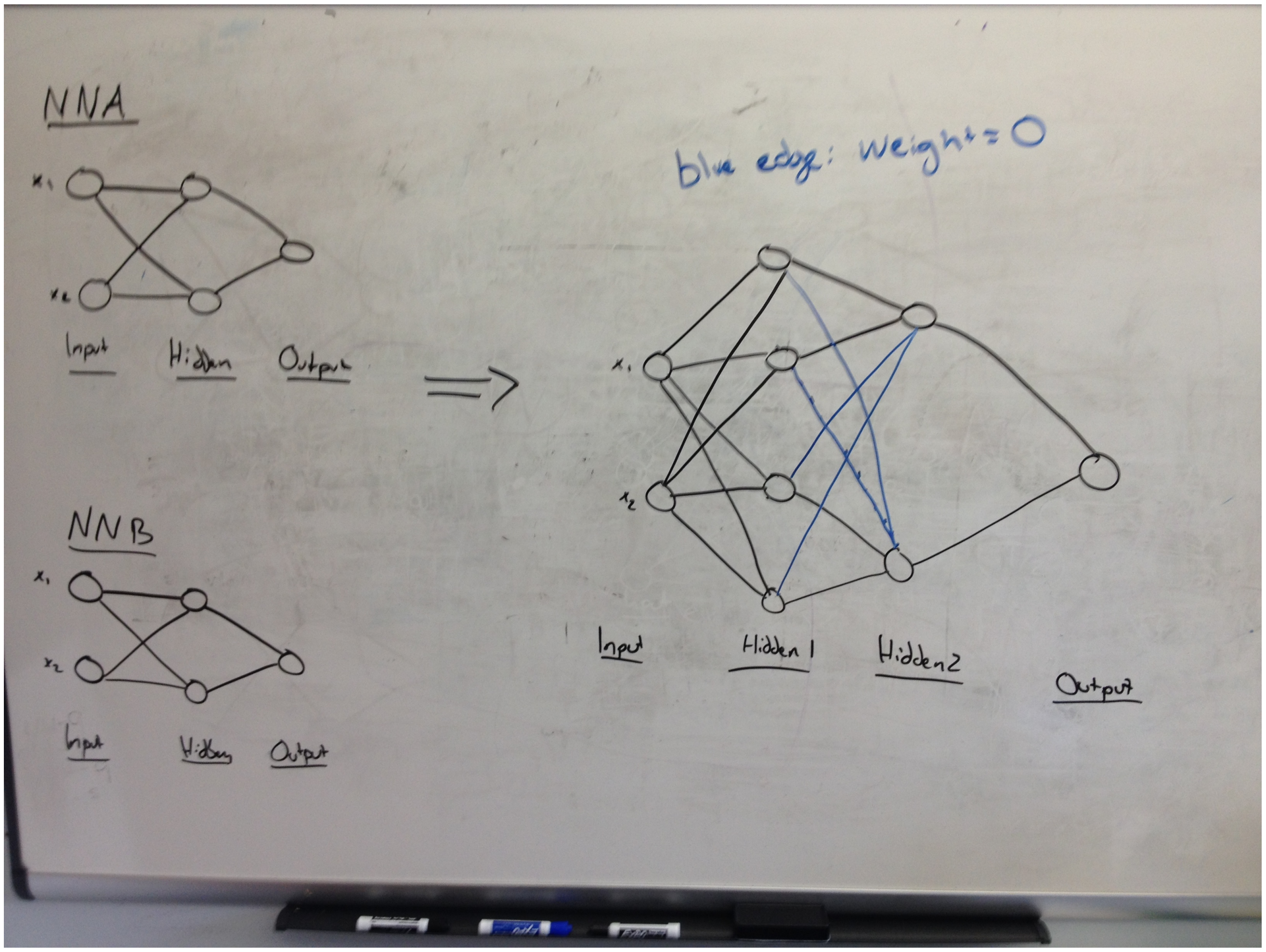


(b) We can generalize the method from a to convert any decision tree to a 2-layer NN. We know from class that any boolean function can be represented by a NN with one hidden layer, although the size of that layer may grow exponentially with the number of variables in the boolean expression. We can represent any tree as a boolean expression by representing each leaf node as a conjunction of all paths taken from each node. We can then combine these leafs by forming a disjunction of each leaf's conjunction, finally forming a single boolean expression.

Once we have a boolean expression, we can create a 2-layer NN. The input layer will have one node for each variable in the boolean expression, and the hidden layer will consist of nodes where each node represents one of the conjunctions from the boolean expression. The weights and biases can then be assigned by hand or trained through back-propagation.

(c) We can construct a single NN for each decision tree as described in (b). We can combine all the NNs into a single NN by treating each output layer as a second hidden layer, where the weights for all incoming edges to the second hidden layer are 0, except for the original edges coming from that output node's corresponding NN. These are represented by the blue edges in the diagram below.

With setup, the result of each decision tree is input into the final output layer, either a +1 or -1, and the majority will be determined by the sign of the output. That is, a negative final output signifies that the majority of the individual NNs (and thus decision trees) calculated a negative result. See the diagram below for clarification.


NNA
x1
x2
Input
Hidden
Output
NNB
x1
x2
Input
Hidden
Output
blue edge: weight = 0
x1
x2
Input
Hidden 1
Hidden2
Output

3. We know that to train a classifier using logistic regression, we can use gradient descent. So the weights are updated as follows:

$$w_i \leftarrow w_i - \mu \frac{\partial \mathcal{L}}{\partial w_i} = w_i - \mu \sum_{j=0}^{||D||} \frac{y^{(j)}}{1 + e^{(wx^{(j)}+b)y^{(j)}}} x_i$$

Our goal is to re-write this in terms of α. We can define α as the following:

$$\alpha_j = \frac{y^{(j)}}{1 + e^{(wx^{(j)}+b)y^{(j)}}}$$

For each α_j, we can now redefine the $wx + b$ terms. For example, for α_1 and some kernel function K, we now have

$$\begin{aligned} wx^{(1)} + b &= \left(\sum_{i=1}^{||D||} \alpha_i \phi(x)^{(i)} y^{(i)} \right) \phi(x)^{(1)} + b \\ &= \left(\sum_{i=1}^{||D||} \alpha_i \phi(x)^{(i)} \phi(x)^{(1)} y^{(i)} \right) + b \\ &= \left(\sum_{i=1}^{||D||} \alpha_i K(x^{(i)}, x^{(1)}) y^{(i)} \right) + b \end{aligned}$$

We now have the original update equation in terms of α instead of in terms of w. So we can rewrite our update function as follows:

$$\alpha_i \leftarrow \alpha_i - \mu \sum_{j=1}^{||D||} \frac{y^{(j)}}{1 + exp\left(((\sum_{k=1}^{||D||} \alpha_k K(x^{(k)}, x^{(j)}) y^{(k)}) + b) y^{(j)} \right)} x_i$$

4. From the hint we have,

$$p(Y = i|x, w_1, \ldots, w_C, b_1, \ldots, b_c) = \frac{e^{w_i x + b_i}}{\sum_{k=1}^{C} e^{w_k x + b_k}}$$

As with 2-class logistic regression, the multiclass closed form can be written as follows,

$$\mathcal{L}(w) = \log \prod_{(x,y)\in D} p(Y = i|x, w_1, \ldots, w_C, b_1, \ldots, b_c))$$

We can rewrite $\mathcal{L}(w)$ as follows.

$$\begin{aligned}
\mathcal{L}(w) &= \log \prod_{(x,y)\in D} p(Y = i|x, w_1, \ldots, w_C, b_1, \ldots, b_c)) \\
&= \sum_{(x,y)\in D} \log(p(Y = i|x, w_1, \ldots, w_C, b_1, \ldots, b_c))) \\
&= \sum_{(x,y)\in D} \log(\frac{e^{w_i x + b_i}}{\sum_{k=1}^{C} e^{w_k x + b_k}}) \\
&= \sum_{(x,y)\in D} \left(\log(e^{w_i x + b_i}) - \log \sum_{k=1}^{C} e^{w_k x + b_k} \right) \\
&= \sum_{(x,y)\in D} \left(w_i x + b_i - \log \sum_{k=1}^{C} e^{w_k x + b_k} \right)
\end{aligned}$$

To find the gradient, we can take the derivative with respect to w_i.

$$\frac{\partial \mathcal{L}}{\partial w_i} = \sum_{(x,y)\in D} \left(x - \frac{x e^{w_i x + b_i}}{\sum_{k=1}^{C} e^{w_k x + b_k}} \right)$$

5. (a) We have the following for the probability density function of a Poisson Distribution,

$$p(X = x|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

We can use the likelihood estimation technique to estimate the parameter λ for N days.

$$\begin{aligned}
\mathcal{L}(\lambda) &= \log \prod_{i=1}^{N} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \\
&= \sum_{i=1}^{N} \log(\frac{\lambda^{x_i} e^{-\lambda}}{x_i!}) \\
&= \sum_{i=1}^{N} \big(\log(\lambda^{x_i} e^{-\lambda}) - \log(x_i!)\big) \\
&= \sum_{i=1}^{N} \big(\log(\lambda^{x_i}) + \log(e^{-\lambda}) - \log(x_i!)\big) \\
&= \sum_{i=1}^{N} (x_i \log(\lambda) - \lambda - \log(x_i!))
\end{aligned}$$

To find the maximum likelihood, we can take the derivative of L.

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \lambda} &= \sum_{i=1}^{N} \Big(\frac{x_i}{\lambda} - 1\Big) \\
&= \sum_{i=1}^{N} \Big(\frac{x_i}{\lambda}\Big) - N \\
&= \frac{x_1 + \cdots + x_n}{\lambda} - N
\end{aligned}$$

Setting $\frac{\partial \mathcal{L}}{\partial \lambda} = 0$, we get

$$\begin{aligned}
0 &= \frac{x_1 + \cdots + x_n}{\lambda} - N \\
N &= \frac{x_1 + \cdots + x_n}{\lambda} \\
N\lambda &= x_1 + \cdots + x_n \\
\lambda &= \frac{x_1 + \cdots + x_n}{N}
\end{aligned}$$

(b) With the given input, we have

$$\lambda = \frac{x_1 + \cdots + x_n}{N} = \frac{72}{10} = 7.2$$

So the probability of seeing only 3 red cars would be

$$p(X = 3|7.2) = \frac{7.2^3 e^{-7.2}}{3!} = \frac{0.28}{6} = 0.046$$