STA 561 HW3 (Convexity, Logistic, Boosting)

Daniel Truver

2/7/2018

(1) Linear Separability Implies Disjoint Convex Hulls

We proceed with proof by contradiction.

For $\{x_n\}$ and $\{x'_m\}$, let their convex hulls be given by

$$X = \{x : x = \sum_n \alpha_n x_x;\ \alpha_n \geq 0;\ \sum_n \alpha_n = 1\}$$
$$X' = \{x : x = \sum_m \alpha'_m x'_m;\ \alpha'_m \geq 0;\ \sum_m \alpha'_m = 1\}$$

(RAA) Suppose $\{x_n\}$ and $\{x'_m\}$ are linearly separable such that for a vector w and scalar w_0 $w^T x_n + w_0 > 0,\ w^T x'_m + w_0 < 0\ \forall x_n, x'_m$ and $X \cap X' \neq \emptyset$.

Then

$$\exists\ x \in X \cap X'$$

such that

$$x = \sum_n \alpha_n x_n = \sum_m \alpha'_m x'_m \tag{1}$$
$$w^T x = w^T \sum_n \alpha_n x_n = w^T \sum_m \alpha'_m x'_m \tag{2}$$
$$w^T x = \sum_n \alpha_n (w^T x_n) = \sum_m \alpha'_m (w^T x'_m) \tag{3}$$

But, from linear separability, we have $w^T x_n > -w_0$ and $w^T x'_m < -w_0$. If we use these relations in the summations above, we obtain

$$\sum_n \alpha_n (w^T x_n) > \sum_n \alpha_n (-w_0), \quad \sum_m \alpha'_m (w^T x'_m) < \sum_m \alpha'_m (-w_0)$$
$$\implies \sum_n \alpha_n (w^T x_n) > -w_0 \sum_n \alpha_n, \quad \sum_m \alpha'_m (w^T x'_m) < -w_0 \sum_m \alpha'_m$$
$$\implies \sum_n \alpha_n (w^T x_n) > -w_0, \quad \sum_m \alpha'_m (w^T x'_m) < -w_0$$
$$\implies w^T x > -w_0 \quad \text{and} \quad w^T x < -w_0 \quad \text{by (3)}$$
$$\Rightarrow\!\Leftarrow$$

Therefore, X and X' are disjoint; the convex hulls do not intersect.

---

(2) Logistic Regression and Gradient Descent

(a) Logistic Sigmoid Function

$$\begin{aligned}
\sigma'(a) &= \frac{d}{da}(1+e^{-a})^{-1} \\
&= e^{-a}(1+e^{-a})^{-2} \\
&= (1+e^{-a})^{-1}\left(\frac{e^{-a}}{1+e^{-a}}\right) \\
&= \sigma(a)\left(\frac{1+e^{-a}-1}{1+e^{-a}}\right) \\
&= \sigma(a)\left(1-\frac{1}{1+e^{-a}}\right) \\
&= \sigma(a)(1-\sigma(a))
\end{aligned}$$

(b) Derivative of Cross Entropy

$$\begin{aligned}
\frac{\partial}{\partial w_j}L_w &= \sum_{i=1}^{n}\frac{\partial}{\partial w_j}\left(-y^{(i)}\log\sigma(w^Tx^{(i)})\right) - \frac{\partial}{\partial w_j}\left((1-y^{(i)})\log(1-\sigma(w^Tx^{(i)}))\right) \\
&= \sum_{i=1}^{n} -y^{(i)}\frac{1}{\sigma(w^Tx^{(i)})}\sigma(w^Tx^{(i)})(1-\sigma(w^Tx^{(i)}))(-x_j^{(i)}) \\
&\qquad - (1-y^{(i)})\frac{1}{1-\sigma(w^Tx^{(i)})}\left(-\sigma(w^Tx^{(i)})\right)\left(1-\sigma(w^T)\right)(-x_j^{(i)}) \\
&= \sum_{i=1}^{n} y^{(i)}x_j^{(i)}\left(1-\sigma(w^Tx^{(i)})\right) - (1-y^{(i)})x_j^{(i)}\sigma(w^Tx^{(i)}) \\
&= \sum_{i=1}^{n} x_j^{(i)}\left(y^{(i)}\left(1-\sigma(w^Tx^{(i)})\right) - (1-y^{(i)})\sigma(w^Tx^{(i)})\right)
\end{aligned}$$

(c) Convexity of the Cross Entropy Loss

We want to show that, for $x, z \in \mathbb{R}^p$ and $\theta \in [0,1]$,

$$L_w(\theta x + (1-\theta)z) \leq \theta L(x) + (1-\theta)L(z).$$

It suffices to show that $\forall i$,

$$g(x^{(i)}) = -y^{(i)}\log\sigma(w^Tx^{(i)}) - (1-y^{(i)})\log(1-\sigma(w^Tx^{(i)}))$$

is convex.

We consider 2 cases, $y^{(i)} = 1$ and $y^{(i)} = 0$. Case 1 ($y^{(i)} = 1$):

$$g(x) = -\log\sigma(w^Tx) = \log(1+e^{-w^Tx})$$

If we take the second derivative of this function in any direction x_j, we get

$$\begin{aligned}\frac{\partial^2}{\partial^2 x_j} g(x) &= \frac{\partial}{\partial x_j}\left(-w_j e^{-w^T x}(1+e^{-w^T x})^{-1}\right) \\ &= w_j^2 e^{-w^T x}(1+e^{-w^T x})^{-1} - w_j^2 (e^{-w^T x})^2 (1+e^{-w^T x})^{-2} \\ &= w_j^2 e^{-w^T x}(1+e^{-w^T x})^{-1}(1-e^{-w^T x}(1+e^{-w^T x})^{-1}) \\ &\geq 0\end{aligned}$$

Therefore, this function is convex.

Case 2 ($y^{(i)} = 0$):

$$f(x) = -\log(1-\sigma(w^T x))$$

Taking the second derivative w.r.t. x_j, we get

$$\begin{aligned}\frac{\partial^2}{\partial^2 x_j} f(x) &= \frac{\partial^2}{\partial^2 x_j}\left(-\log(e^{-w^T x}) - \log(1+e^{-w^T x})^{-1}\right) \\ &= \frac{\partial}{\partial x_j}\left(w_j - w_j e^{-w^T x}(1+e^{-w^T x})^{-1}\right) \\ &\geq 0\end{aligned}$$

since this derivative will be the same as Case 1.

Therefore, we have shown the summand of the cross entropy loss function to be complex $\forall i$. The positive sum of convex functions is convex, so the cross entropy loss is convex.

(d) See Code

---

(3) Boosting

(a) Perfect Training Classification

Assume the the weak learning assumption (WLA). So, we get

$$\epsilon_t = \text{error}_{d_t}(h_{(t)}) =: \frac{1}{2} - \gamma_t, \quad \gamma_t > \gamma_{WLA}$$

Now sit and stare at the problem for a couple hours. Get frustrated. Eat a sandwich. Come back to the problem. Write down thought process.

```
##
## All work and no play makes Jack a dull boy.
## All work and no play makes Jack a dull boy.
## All work and no play makes Jack a dull boy.
## All work and no play makes Jack a dull boy.
## All work and no play makes Jack a dull boy.
```

Another sandwich. Scroll up in the lecture notes. Epiphany.

$$
\begin{aligned}
\text{classification error} &\leq \frac{1}{n}\sum_{i=1}^{n} e^{-y_i f(x_i)} \\
&= \frac{1}{n}\sum_{i=1}^{n} \exp\left(-y_i \sum_{t=1}^{T} \alpha_t h_{(t)}(x_i)\right) \\
&= \frac{1}{n}\sum_{i=1}^{n} \prod_{t=1}^{T} \exp\left(-y_i \alpha_t h_{(t)}(x_i)\right) \\
&= \sum_{i=1}^{n} \frac{1}{n} \exp\left(-y_i \alpha_1 h_{(1)}(x_i)\right) \cdots \exp\left(-y_i \alpha_T h_{(T)}(x_i)\right) \\
&= \sum_{i=1}^{n} d_{1,i} \exp\left(-y_i \alpha_1 h_{(1)}(x_i)\right) \cdots \exp\left(-y_i \alpha_T h_{(T)}(x_i)\right) \\
&= \sum_{i=1}^{n} d_{2,i} Z_1 \exp\left(-y_i \alpha_2 h_{(2)}(x_i)\right) \cdots \exp\left(-y_i \alpha_T h_{(T)}(x_i)\right) \\
&= \sum_{i=1}^{n} d_{T+1,i} (Z_1 \cdots Z_T) \\
&= \prod_{t=1}^{T} Z_t \qquad (d_{T+1} \text{ is a distribution})
\end{aligned}
$$

We now have the error bounded by the product of the normalization constants which appear to be functions of t. We now want to find a bound of each indivual Z_t, hopefully still a function of t, and show that everything together approaches 0 as t approaches ∞.

$$
\begin{aligned}
Z_t &= \sum_{i=1}^{n} d_{t,i} e^{-y_i \alpha_t h_{(t)}(x_i)} \\
&= \sum_{i:h_t(x_i)=y_i} d_{t,i} e^{-\alpha_t} + \sum_{i:h_t(x_i)\neq y_i} d_{t,i} e^{\alpha_t} \\
&= (\gamma_t + 1/2) e^{-\alpha_t} + (1/2 - \gamma_t) e^{\alpha_t} \qquad \text{(by WLA)} \\
&= (\gamma_t + 1/2) \left(\frac{1/2 - \gamma_t}{\gamma_t + 1/2}\right)^{\frac{1}{2}} + (1/2 - \gamma_t) \left(\frac{\gamma_t + 1/2}{1/2 - \gamma_t}\right)^{\frac{1}{2}} \quad \text{(by defn. of } \alpha_t) \\
&= 2\sqrt{(\gamma_t + 1/2)(1/2 - \gamma_t)} \\
&= \sqrt{1 - 4\gamma_t^2} \\
&< \sqrt{1 - 4\gamma_{WLA}^2} \qquad (\gamma_t > \gamma_{WLA}) \\
&= r < 1 \qquad \left(\gamma_{WLA} \in (0, \frac{1}{2})\right)
\end{aligned}
$$

This gives us

$$\text{training error} < \prod_{t=1}^{T} Z_t < \prod_{t=1}^{T} r = r^T$$

Since $r < 1$, we can view r^T as a monotonically decreasing sequence in T. Given $\epsilon > 0$, take

$$T > \frac{\log \epsilon}{\log r}$$

Then we have

$$T \log r < \log \epsilon \implies r^T < \epsilon$$

Bringing it all together, we get $\forall \epsilon > 0, \exists\, T$ such that.

$$\text{training error} < \epsilon$$

(b) Weighted Points in Training Set

Suppose that our objective is

$$R^{train}(\lambda) = \sum_{i=1}^{n} w_i e^{-(M\lambda)_i}$$

It seems that the standard AdaBoost we already derived is just a special case of this where all weights are $w_i = 1/n$. For this reason, the derivation and results should look similar. We are still going to look for an optimal direction and step size in that direction to iteratively reduce the objective above. That is, we are still doing coordinate descent, just with more subscripts to track.

$$\begin{aligned}
j_t &\in \text{argmax}_j \left[-\frac{\partial R^{train}(\lambda_t + \alpha \mathbf{e}_j)}{\partial \alpha}\bigg|_{\alpha=0} \right] \\
&= \text{argmax}_j \left[-\frac{\partial}{\partial \alpha} \left[\sum_{i=1}^{n} w_i e^{-(M(\lambda_t + \alpha e_j))_i} \right] \bigg|_{\alpha=0} \right] \\
&= \text{argmax}_j \left[\sum_{i=1}^{n} w_i M_{ij} e^{-(M\lambda_t)_i} \right] \quad \text{(some algebra later)}
\end{aligned}$$

This time, our discrete probability distribution will be

$$d_{t,i} = w_i e^{-(M\lambda_t)_i} / Z_t \text{ where } Z_t = \sum_{i=1}^{n} w_i e^{-(M\lambda_t)_i}$$

Same as before, the argmax does not change when we multiply by Z_t and we have absorbed the w_i into our distribution, so we are left with

$$j_t \in \text{argmax}_j \sum_{i=1}^{n} M_{ij} d_{t,i} = \text{argmax}(d_t^T M)_j.$$

Our caculation of the step size is also fairly similar. We search along the direction j_t by setting the derivative equal to 0.

$$\begin{aligned}
0 &= \frac{\partial R^{train}(\lambda_t + \alpha e_{j_t})}{\partial \alpha}\bigg|_{\alpha_t} \\
&= -\sum_{i=1}^{n} w_i M_{ij_t} e^{-(M\lambda_t)_i - \alpha_t M_{ij_t}} \\
&= -\sum_{i: M_{ij_t}=1} w_i e^{-(M\lambda_t)_i} e^{-\alpha_t} - \sum_{i: M_{ij_t}=-1} w_i - e^{(M\lambda_t)_i} e^{\alpha_t}
\end{aligned}$$

Now we multiply by $-1/Z_t$ to get us one step closer to the end of our suffering:

$$
\begin{aligned}
0 &= \sum_{i:M_{ij_t}=1} d_{t,i} e^{-\alpha_t} - \sum_{i:M_{ij_t}=-1} d_{t,i} e^{\alpha_t} \\
&=: d_+ e^{-\alpha_t} - d_- e^{\alpha_t} \\
d_- e^{\alpha_t} &= d_+ e^{-\alpha_t} \\
\alpha_t &= \frac{1}{2} \ln \frac{1 - d_-}{d_-}
\end{aligned}
$$

So, in the end, the coordinate descent algorithm is... exactly the same. The only difference is that

$$d_{1,i} = w_i \text{ for } i = 1, \ldots, n.$$

Now, to justify this result to myself, I'll walk through my intuition.

The idea of the AdaBoost algorithm is to iteratively reduce the training error by up-weighting misclassified points and down-weighting correctly classified points. We do not want to substantially alter this behavior. Instead, we set the initial importance of the points by choosing our own weights. Points that we consider important will start off with much more weight in the algorithm and will keep that extra weight through several iterations (how many iteration is context dependent). This additional weight in the beginning will make our algorithm prioritize the highly weighted points and try to classfy them correctly from the start. I suspect this is useful when we have imbalanced data.