Homework 3: Ensemble Methods and Recurrent Neural Networks

Large-Scale Data Analysis 2020

Name and KU ID (ABC123)

May 23, 2020

1 AdaBoost (?? pts.)

Let $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\} \in (\mathbb{R}^d \times \{-1, 1\})^n$ be the training data, $t = 1, \ldots, T$ denote the boosting rounds. At round t, $w_i^{(t)}$ is the weight of training pattern i, $h^{(t)}$ is the base classifier from round t, $\alpha^{(t)}$ is the importance of this classifier, $\varepsilon^{(t)} = \sum_{j=1}^{n} w_j \mathbb{I}(h^{(t)}(\boldsymbol{x}_j) \neq y_j)$, and the aggregated classifier is $f^{(t)} = \sum_{i=1}^{t} \alpha^{(i)} h^{(i)}$.

1.1 Step 1

The goal is to prove for each i at any round t

$$w_i^{(t+1)} = w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{2\sqrt{\varepsilon^{(t)}\left(1 - \varepsilon^{(t)}\right)}} \quad . \tag{1}$$

We start from the update of the weights as defined in the lecture:

$$w_i^{(t+1)} = \begin{cases} w_i^{(t)} / \left(2\varepsilon^{(t)}\right) & \text{if } h^{(t)}(\boldsymbol{x}_j) \neq y_j \\ w_i^{(t)} / \left(2\left(1 - \varepsilon^{(t)}\right)\right) & \text{otherwise} \end{cases} \tag{2}$$

Roughly speaking, at each step, we want the weights of miss-classified training samples to increase, and the weights of the correctly classified ones to decrease. So, given that $\varepsilon^{(t)} < 0.5$, the weights $w_i^{(t)}$ are multiplied by $\exp(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i)$, since:

$$\exp(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i) = \begin{cases} \exp\left[-\alpha^{(t)}\right] < 1 & \text{if } (t)\,(\boldsymbol{x}_i) = y_i \\ \exp\left[\alpha^{(t)}\right] > 1 & \text{if } h^{(t)}\,(\boldsymbol{x}_i) \neq y_i \end{cases} \tag{3}$$

Also, since we want that the weights, at any round t, are normalized such that $\sum_{i=1}^{n} w_i^{(t)} = 1$, we divide the updated weight by $\sum_{i=}^{n} w_i^{(t)} \exp(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i)$ to obtain:

$$w_i^{(t+1)} = w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{\sum_{i=}^{n} w_i^{(t)} \exp(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i)} \tag{4}$$

Rearranging the normalizer at the denominator, substituting terms from equation (2) and given that $\alpha^{(t)} = \frac{1}{2}\ln\left(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}\right)$, it follows that:

$$
\begin{aligned}
w_i^{(t+1)} &= w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{\sum_{i:y_i=h^{(t)}(x_i)} w_i^{(t)} \exp(-\alpha^{(t)}) + \sum_{i:y_i\neq h_t(x_i)} w_i^{(t)} \exp(\alpha^{(t)})} \\
&= w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{(1-\varepsilon^{(t)})\exp(-\alpha^{(t)}) + \varepsilon \exp(\alpha^{(t)})} \\
&= w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{(1-\varepsilon^{(t)})\exp(-\frac{1}{2}\ln(\frac{1-\varepsilon}{\varepsilon})) + \varepsilon \exp(\frac{1}{2}\ln(\frac{1-\varepsilon}{\varepsilon}))} \\
&= w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}\left(\boldsymbol{x}_i\right) y_i\right)}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}}
\end{aligned}
\tag{5}
$$

In the previous steps, I showed how equation (2) can be rewritten as equation (1), now I show the prof that equation (1) is valid for both wrongly classified and correctly classified data points.

For wrongly classified data points $h^{(t)}(\boldsymbol{x}_i)y_i = -1$, substituting this equivalence to equation (1) I obtain:

$$
\begin{aligned}
w^{(t+1)} &= w_i^{(t)} \frac{\exp(\alpha)}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} \\
&= w_i^{(t)} \frac{\exp(\frac{1}{2}\ln(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}))}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} = \frac{w_i^{(t)}}{2\varepsilon}
\end{aligned}
\tag{6}
$$

For correctly classified data points $h^{(t)}(\boldsymbol{x}_i)y_i = 1$, substituting this equivalence to equation (1) I obtain:

$$
\begin{aligned}
w^{(t+1)} &= w_i^{(t)} \frac{\exp(-\alpha)}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} \\
&= w_i^{(t)} \frac{\exp(-\frac{1}{2}\ln(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}))}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} \\
&= \frac{w_i^{(t)}}{\exp(\frac{1}{2}\ln(\frac{1-\varepsilon^{(t)}}{\varepsilon^{(t)}}))2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} = \frac{w_i^{(t)}}{2(1-\varepsilon)}
\end{aligned}
\tag{7}
$$

1.2 Step 2

The goal is to prove for each i at any round t

$$
w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} = \frac{\exp\left(-f^{(t)}(\boldsymbol{x}_i) y_i\right)}{n\prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(\tau)}\left(1-\varepsilon^{(\tau)}\right)}} \quad . \tag{8}
$$

The weights on the data points can be computed recursively, and at each iteration they are normalized by $2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}$, as shown in the following:

$$
\begin{aligned}
w^{(t+1)} &= w_i^{(t)} \frac{\exp\left(-\alpha^{(t)} h^{(t)}(\boldsymbol{x}_i) y_i\right)}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)}} \\
&= w_i^{(t-1)} \frac{\exp(-y_i(\alpha^{(t)} h^{(t)}(x_i) + \alpha^{(t-1)} h^{(t-1)}(x_i))}{2\sqrt{\varepsilon^{(t)}\left(1-\varepsilon^{(t)}\right)} 2\sqrt{\varepsilon^{(t-1)}\left(1-\varepsilon^{(t-1)}\right)}} \\
&= .... \\
&= \frac{\exp(-y_i \sum_{t=1}^{t} \alpha^{(t)} h^{(t)}(\boldsymbol{x}))}{n \prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(\tau)}\left(1-\varepsilon^{(\tau)}\right)}}
\end{aligned} \tag{9}
$$

Since the decision function $f(t) = \sum_{t=1}^{t} \alpha^{(t)} h^{(t)}$, it follows that **(check if t=1 or i=1, and the exponents of alpha and h, also n??)**:

$$
\frac{\exp(-y_i \sum_{t=1}^{t} \alpha^{(t)} h^{(t)}(\boldsymbol{x}))}{n \prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(\tau)}\left(1-\varepsilon^{(\tau)}\right)}} = \frac{\exp\left(-f^{(t)}\left(\boldsymbol{x}_i\right) y_i\right)}{n \prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(\tau)}\left(1-\varepsilon^{(\tau)}\right)}} \tag{10}
$$

1.3 Step 3

The goal is to prove for each i at any round t

$$
\frac{\exp\left(-f^{(t)}(\boldsymbol{x}_i) y_i\right)}{n \prod_{\tau=1}^{t} 2\sqrt{\varepsilon^{(\tau)}\left(1-\varepsilon^{(\tau)}\right)}} = \frac{\exp\left(-f^{(t)}(\boldsymbol{x}_i) y_i\right)}{\sum_{l=1}^{n} \exp\left(-f^{(t)}(\boldsymbol{x}_l) y_l\right)} \quad . \tag{11}
$$

The weights are normalized, that is, ...

2 Gradient Boosting

3 Recurrent Neural Networks

Replace the simple recurrent layer in `LSDA2020_RNN1.ipynb` notebook with a LSTM layer. Follow the equations from `LSDA2020_RNN2.ipynb` to implement the LSTM. You will need to add more variables and extend the step function. Also remember that the LSTM not only transfers the hidden state h_t to the next time step $t+1$, but also the cell state c_t.

4 Recurrent Neural Networks in Keras

Add different components to the RNN from `LSDA2020_RNN2.ipynb` and report the results on the validation set (the changed parts in the code).

1. Add bidirectional sequence processing by utilizing `tf.keras.layers.Bidirectional`.
2. Stack 2 LSTM layers. What is the difference to bidirectional processing? (you may need to use the `return_sequences` parameter)
3. Add a 1-d convolution layer (`tf.keras.layers.Conv1D`) before the recurrent part. You will need to reshape the data, you can use the `tf.keras.layers.Reshape` layer for that.
4. Gradient clipping can be a helpful to train recurrent networks. Keras offers to clip gradients directly through the optimizer. Try this with clip values of 0.1, 1, and 10.