Homework 2

Chen Lu

April 23, 2018

1 Boosting: From Weak to Strong

1. For each threshold s there is some $m_0(s) \in 0, 1, ..., m$ such that

$$x^i \geq s \quad i \in \{1, ..., m_0(s)\} \tag{1.1}$$
$$x^i < s \quad i \in \{m_0(s)+1, ..., m\} \tag{1.2}$$

that is

$$\varphi_{s,+}(x^{(i)}) \begin{cases} 1 & i \in \{0, 1, ..., m_0(s)\} \\ -1 & i \in \{m_0(s)+1, ..., m\} \end{cases} \tag{1.3}$$

Therefore,

$$\begin{aligned}
\sum_{i=1}^{m} p_i 1\{\varphi_{s,+}(x^{(i)}) \neq y^{(i)}\} &= \sum_{i=1}^{m_0(s)} p_i 1\{y^{(i)} = -1\} + \sum_{i=m_0(s)+1}^{m} p_i 1\{y^{(i)} = 1\} && (1.4)\\
&= \sum_{i=1}^{m_0(s)} p_i \frac{1-y^{(i)}}{2} + \sum_{i=m_0(s)+1}^{m} p_i \frac{1+y^{(i)}}{2} && (1.5)\\
&= \sum_{i=1}^{m_0(s)} \frac{p_i}{2} - \sum_{i=1}^{m_0(s)} \frac{p_i y^{(i)}}{2} + \sum_{i=m_0(s)+1}^{m} \frac{p_i}{2} + \sum_{i=m_0(s)+1}^{m} \frac{p_i y^{(i)}}{2} && (1.6)\\
&= \frac{\sum_{i=1}^{m} p_i}{2} - \frac{1}{2}(\sum_{i=1}^{m_0(s)} y^{(i)} p_i - \sum_{i=m_0(s)+1}^{m} y^{(i)} p_i) && (1.7)\\
&= \frac{1}{2} - \frac{1}{2}(\sum_{i=1}^{m_0(s)} y^{(i)} p_i - \sum_{i=m_0(s)+1}^{m} y^{(i)} p_i) && (1.8)
\end{aligned}$$

Similarly, we can get

$$\sum_{i=1}^{m} p_i 1\{\varphi_{s,-}(x^{(i)}) \neq y^{(i)}\} = \frac{1}{2} - \frac{1}{2}(\sum_{i=m_0(s)+1}^{m} y^{(i)} p_i - \sum_{i=1}^{m_0(s)} y^{(i)} p_i) \tag{1.9}$$

2. For each $m_0 \in \{0, ..., m\}$

$$f(m_0) = \sum_{i=1}^{m_0(s)} y^{(i)} p_i - \sum_{i=m_0(s)+1}^{m} y^{(i)} p_i$$

We need to show that $\gamma = \frac{1}{2m}$ satisfies that

$$\underset{m_0}{max} |f(m_0)| \geq 2\gamma = \frac{1}{m} \tag{1.10}$$

If we show that

$$|f(m_0) - f(m_0+1)| \geq \frac{2}{m} \tag{1.11}$$

then, we can get (1.10).
Thus I bring the definition of $f(m_0)$ into (1.11)

$$\begin{aligned} |\sum_{i=1}^{m_0(s)} y^{(i)} p_i - \sum_{i=m_0(s)+1}^{m} y^{(i)} p_i - \sum_{i=1}^{m_0(s)+1} y^{(i)} p_i + \sum_{i=m_0(s)+2}^{m} y^{(i)} p_i| &= 2|y^{(m_0+1)} p_{m_0+1}| \qquad (1.12) \\ &= 2|p_{m_0+1}| \qquad (1.13) \end{aligned}$$

There must exist some $m_o + 1$, $p_{m_0+1} \geq \frac{1}{m}$. Otherwise, it can't satisfy the constraint $\sum_{i=1}^{m} p_i = 1$
Now, we have

$$p_{m_0+1} \geq \frac{1}{m} \tag{1.14}$$

bring it into (1.13), get $|f(m_0) - f(m_0+1)| \geq \frac{2}{m}$.
3. From 2., we can get $\gamma = \frac{1}{2m}$.
According to Theorem 1.

$$\begin{aligned} J_t &\leq \sqrt{1-4\gamma^2} J_{t_1} \qquad (1.15) \\ &\leq (\sqrt{1-4\gamma^2})^{t-1} J_1 \qquad (1.16) \\ &\leq (1-4\gamma^2)^{\frac{t-1}{2}} (\frac{1}{2} - \lambda) \qquad (1.17) \end{aligned}$$

Obvioiusly, if $J_t < \frac{1}{m}$, we have zero training error. Thus,

$$J_t \le (1-4\gamma^2)^{\frac{t-1}{2}}(\frac{1}{2}-\lambda) < \frac{1}{m} \tag{1.18}$$

we can derive from (1.18) that

$$\begin{aligned} t &< -\frac{\log m^2 \frac{1-2\gamma}{1+2\gamma}}{\log(1-4\gamma^2)} && (1.19)\\ &= 1-2\frac{\log \frac{m}{1+2\gamma}}{\log(1-4\gamma^2)} && (1.20)\\ &= \frac{\log \frac{4(m+1)}{(m-1)m^2}}{\log(1-\frac{1}{m^2})} && (1.21)\end{aligned}$$

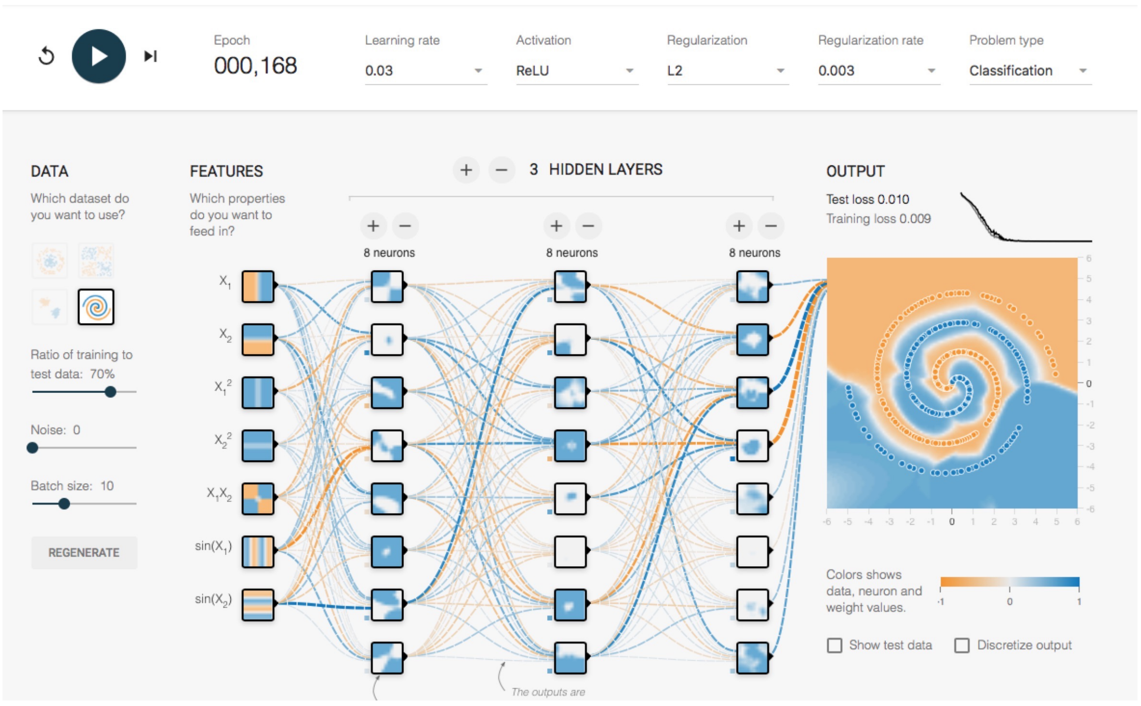
2 Deep Neural Networks

2.1 BEST CONFIGURATION

My best conguration is shown in Figure 2.1. It achieves 0.0009 loss in training dataset and 0.01 loss in testing dataset. I used ReLU as the activation function and the learning rate is set as 0.03. L2 regularization is employed, and regularization rate is 0.003. All features are included in the model and the network size is as big as the website could provide.

2.2 INFLUENCING FACTORS

- learning rate
  Generally, the bigger the learning rate, the quicker the convergence rate. But if the learning rate is too big, the loss will not change smoothly and hence the result is not very good. If the learning rate is too small, the training will cost a lot of time.
- activation function
  For activation functions, ReLU performs better than the rest. It converges faster than the others and the loss is lower. In this classification problem, the increase of hidden layers would give lower loss finally. Besides, more features would also boost the performance. In short, the biggest model is the best.
- number of hidden layers
  As for the number of hidden layers, more hidden layers, higher learning ability the neural network has. But solving a simple problem is unnecessary with a complicate network. However, simple network can't solve a complex distribute dataset. In the other hand, more hidden layers means that you need more time to calculate.
- regularization
  The regularization would slow down the convergence, and this is true for both L1 and L2. If the regularization rate is too high, the final loss would be about 50%, which is no better than random guess.



Epoch 000,168
Learning rate 0.03
Activation ReLU
Regularization L2
Regularization rate 0.003
Problem type Classification

Ratio of training to test data: 70%
Noise: 0
Batch size: 10

Test loss 0.010
Training loss 0.009

Figure 2.1: Best Configuration

2.3 Tuning "principle"

1. dataset & features
   More data are greatly helpful which can prevent the training procedure from overfitting. What's more, selecting different features and combining them show different performance.

2. Different network structure
   In the future, I will want to use a large number of hidden layers and neurons, making the network deeper and wider. A large network means more flexibility and will ïňĄt the data better. Of course, overfitting is always an importance concern.

3. Activation function
   I need to see the change of the objective function as the running goes on to determine whether I should change the learning rate. This is also what I did in this exercise. An inappropriate activation function will cause an unexpected problem, such as gradience vanishing and gradience exploding.

4. Cross-validation
   Over-fitting is a huge issue for neural network training. When I achieve 0 loss in training set, the loss of testing test goes up however. Cross-validation and regularization are necessary tools to avoid over-fitting.

3 Clustering: Mixture of Multinomials

3.1 MLE for multinomial

Take the log likelihood and use Lagrange Multiplier

$$\begin{aligned} L &= \sum_{j=1}^{N} \log P(X_j|\mu) - \lambda(\sum_i \mu_i - 1) && (3.1)\\ &= \sum_j \log n_j! - \sum_j \sum_i \log x_{ji}! + \sum_j \sum_i (x_{ji} \log \mu_i) + \lambda(\sum_i \mu_i - 1) && (3.2)\end{aligned}$$

where $\sum_i x_{ji} = n_j$. Differentiate it with respect to μ_i and λ, we get

$$\begin{aligned} \frac{\sum_j x_{ji}}{\mu_i} - \lambda = 0, \quad i = 1, ..., d && (3.3)\\ \sum_i \mu_i - 1 = 0 && (3.4)\end{aligned}$$

Therefore

$$\begin{aligned} \lambda &= \sum_i \sum_j x_{ji} && (3.5)\\ \mu_i &= \frac{\sum_j x_{ji}}{\sum_i \sum_j x_{ji}} && (3.6)\end{aligned}$$

3.2 EM for mixture of multinomials

I use maximum likelihood method to derive its EM algorithm. Here, we have

$$\begin{aligned} logP(D) &= \sum_{d=1}^{D} logP(d)\\ &= \sum_{d=1}^{D} log\left[\frac{n_d!}{\prod_{w=1}^{W} T_{dw}!} \sum_{k=1}^{K} \pi_k \prod_{w=1}^{W} \mu_{wk}^{T_{dw}}\right] \end{aligned} \qquad (3.7)$$

Then, by using Lagrange multiplier method, we could get

$$L = logP(D) + \lambda\left(1 - \sum_{k=1}^{K} \pi_k\right) + \sum_{k=1}^{K} \eta_k \left(1 - \sum_{w=1}^{W} \mu_{wk}\right) \qquad (3.8)$$

Next, we set the partial derivations to 0 to find the optimal parameters.

$$\begin{aligned}
\frac{\partial L}{\partial \pi_j} &= 0, \quad j = 1, \dots, K \\
\frac{\partial L}{\partial \lambda} &= 0 \\
\frac{\partial L}{\partial \mu_{ij}} &= 0, \quad i = 1, \dots, W \; j = 1, \dots, K \\
\frac{\partial L}{\partial \eta_j} &= 0, \quad j = 1, \dots, K
\end{aligned} \tag{3.9}$$

After making several simplification steps, we could get the algorithm

$$\begin{aligned}
E-step: \quad \gamma_{dj} &= \frac{\pi_j \prod_{w=1}^{W} \mu_{wj}^{T_{dw}}}{\sum_{k=1}^{K} \pi_k \prod_{w=1}^{W} \mu_{wk}^{T_{dw}}} \\
M-step: \quad \pi_j &= \frac{1}{D} \sum_{d=1}^{D} \gamma_{dj} \\
\mu_{ij} &= \frac{\sum_{d=1}^{D} T_{di} \gamma_{dj}}{\sum_{d=1}^{D} n_d \gamma_{dj}}
\end{aligned} \tag{3.10}$$

I implemented this EM algorithm using Python, and tested it on the "20newsgroup" dataset. I tested with different parameters several times, and show one result as follows.

- K = 5: edu, edu, edu, edu, god
- K = 10: god, edu, edu, people, edu, edu, drive, key, com, israel
- K = 20: edu, edu, file, edu, god, edu, window, key, drive, god, god, turkish, gun, com, edu, edu, president, space, key, edu
- K = 30: edu, edu, edu, space, israel, president, jesus, edu, key, window, edu, game, gun, file, people, jpeg, car, players, key, turkish, space, edu, db, god, windows, com, edu, edu, objective, drive

When thinking about choosing the "best" K value, it maybe depend on how many categories we want, whether in coarse grain or fine grain. After considering the most-frequent words in each topic for each case, I thought K = 20 may be a good chose if we want topics divided in fine grain, K = 10 may also be accepted if in coarse grain.
The implementing and debugging procedure was very interesting. Firstly, I removed stop words and filtered vocabulary list using TF-IDF feature. Then, I initialized the model parameters under Dirichlet distribution and calculated the underflow probability value by using the log trick. Furthermore, I adjusted several parameters such as smoothing value, filtered vocabulary size. After this implementation, I earned a deep sight into the EM algorithm and learned a basic knowledge about the topic model.