Neural Networks
From Deep Learning specialization

October 26, 2018

1 Logistic regression

Parameters used in logistic regression :

n_x	:	number of features
$x \in \mathbb{R}^{n_x}$	:	input features vector
$y \in \{0,1\}$	:	training label
$w \in \mathbb{R}^{n_x}$	:	weights
$b \in \mathbb{R}$	:	threshold
$\hat{y} = \sigma(w^T x + b)$	:	the output
$\sigma(z) = \frac{1}{1+\exp(-z)}$	:	sigmoid function
$\mathcal{L}(y,\hat{y}) = -y\log(\hat{y}) - (1-y)\log(1-\hat{y})$	:	cost function (single example)

See Figure 1 for an illustration of the sigmoid function.

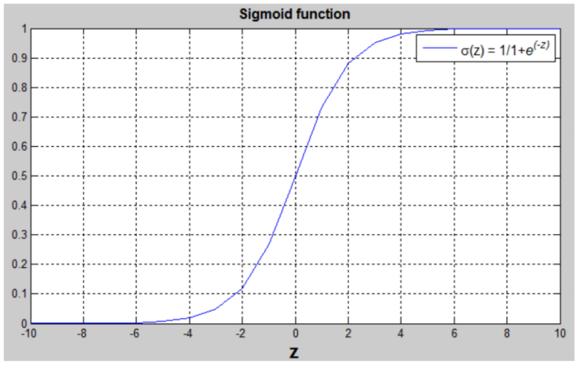

Figure 1: Sigmoid function

Since we have multiple examples, we take the mean over all examples for this cost function. Let m be the number of examples.

$$Y, \hat{Y} \in \mathbb{R}^{1\times m} \tag{1}$$

$$J(Y,\hat{Y}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(Y^{(i)},\hat{Y}^{(i)}) \tag{2}$$

$$= \frac{-1}{m}\sum_{i=1}^{m}\left(Y^{(i)}\log(A^{[L](i)}) + (1-Y^{(i)})\log(1-A^{[L](i)})\right) \tag{3}$$

1.1 Regularization

1.1.1 L_2 regularization

We have $W \in \mathbb{R}^{1\times n_x}$. The cost function becomes as follows.

$$J(Y,\hat{Y}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(Y^{(i)},\hat{Y}^{(i)}) + \frac{\lambda}{2m}||W||_2^2 \tag{4}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(Y^{(i)},\hat{Y}^{(i)}) + \frac{\lambda}{2m}\sum_{k=1}^{n_x}W_k^2 \tag{5}$$

1.1.2 L_1 regularization

The cost function becomes as follows.

$$J(Y,\hat{Y}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(Y^{(i)},\hat{Y}^{(i)}) + \frac{\lambda}{2m}||W||_1 \tag{6}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(Y^{(i)},\hat{Y}^{(i)}) + \frac{\lambda}{2m}\sum_{k=1}^{n_x}|W_k| \tag{7}$$

2 Neural Network

$a^{[i](j)}$ is the hidden unit for the i-th layer and the j-th example.

2.1 Single example

Let $n[i]$ be the number of units in layer i. In particular, we have $n[0] = n_x$.

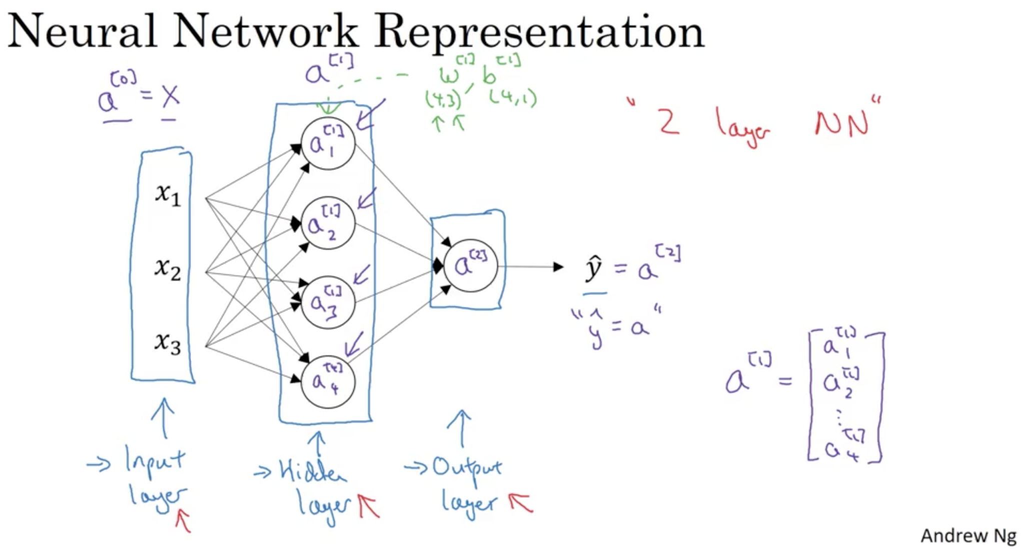

Figure 2: Each node of the neural network corresponds to a different logistic regression computation. For example, $a_1^{[1]} = [w_{11}^{[1]}, w_{12}^{[1]}, w_{13}^{[1]}] \times [x_1, x_2, x_3]^T + b_1^{[1]}$.

$$W^{[i]} \in \mathbb{R}^{n[i] \times n[i-1]}, \quad \forall i \in \{1, \dots, L\} \tag{8}$$
$$a^{[i]}, z^{[i]}, b^{[i]} \in \mathbb{R}^{n[i] \times 1} \tag{9}$$
$$a^{[0]} = x \tag{10}$$
$$z^{[i]} = W^{[i]} a^{[i-1]} + b^{[i]} \tag{11}$$
$$a^{[i]} = g^{[i]}(z^{[i]}) \tag{12}$$
$$\hat{y} = a^{[L]} \tag{13}$$

Where $g^{[i]}(z^{[i]})$ is the activation function of layer i. There is a total of L hidden layers.

In binary classification, we have $y, \hat{y} \in \{0, 1\}$. Usually, we use the same cost function as presented for the logistic regression to compute the error between the true label y and the prediction $\hat{y}$.

2.2 Multiple examples

$$W^{[i]} \in \mathbb{R}^{n[i] \times n[i-1]}, \quad \forall i \in \{1, \dots, L\} \tag{14}$$
$$A^{[i]}, Z^{[i]}, B^{[i]} \in \mathbb{R}^{n[i] \times m} \tag{15}$$
$$X \in \mathbb{R}^{n_x \times m} \tag{16}$$
$$B^{[i]} = [b^{[i]}, \dots, b^{[i]}] \tag{17}$$
$$A^{[0]} = X \tag{18}$$
$$Z^{[i]} = W^{[i]} A^{[i-1]} + B^{[i]} \tag{19}$$
$$A^{[i]} = g^{[i]}(Z^{[i]}) \tag{20}$$
$$\hat{Y} = A^{[L]} \tag{21}$$

Since we have multiple examples, we take the mean over all examples for this cost function. Let m be the number of examples.

$$Y, \hat{Y} \in \mathbb{R}^{1 \times m} \tag{22}$$
$$J(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(Y^{(i)}, \hat{Y}^{(i)}) \tag{23}$$
$$= \frac{-1}{m} \sum_{i=1}^{m} \Big(Y^{(i)} \log(A^{[L](i)}) + (1 - Y^{(i)}) \log(1 - A^{[L](i)}) \Big) \tag{24}$$

2.3 Error correction

The first forward propagation pass does not produce the best possible results. To minimize the cost function $J(Y, \hat{Y})$, we need backward propagation to use the cost of step i to modify weights of step $i + 1$.

Let E be the number of epochs, meaning the number of iterations over all examples of the dataset.

Algorithm 1: Training algorithm with forward and backward propagation

```
Initialize weights $W^{[i]}$ and $b^{[i]}$ for $i \in \{1, \dots, L\}$ ;
for $i = 1..E$ do
    Forward propagation to compute $\hat{Y}$;
    Compute cost $J(Y, \hat{Y})$ ;
    Backpropagation to compute the deltas of weights ;
    Update the weights using $\theta = \theta - \alpha \frac{\delta J}{\delta \theta}$ ;
end
```

2.4 Activation functions

There are a few possibilities for the activation function of each layer of the neural network. Here are the most popular in the litterature.

sigmoid	$g(Z^{[l]}) = \frac{1}{1+\exp^{-Z^{[l]}}}$	$g'(Z^{[l]}) = g(Z^{[l]})(1 - g(Z^{[l]}))$
tanh	$g(Z^{[l]}) = \tanh(Z^{[l]}) = \frac{\exp(Z^{[l]})-\exp(-Z^{[l]})}{\exp(Z^{[l]})+\exp^{-Z^{[l]}}}$	$g'(Z^{[l]}) = 1 - (\tanh(Z^{[l]}))^2$
ReLU	$g(Z^{[l]}) = \max(0, Z^{[l]})$	$g'(Z^{[l]}) = \begin{cases} 0 & \text{if } Z^{[l]} < 0 \\ 1 & \text{if } Z^{[l]} \geq 0 \end{cases}$

2.5 Regularization

$$J(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(Y^{(i)}, \hat{Y}^{(i)}) + \frac{\lambda}{2m} \sum_{j=1}^{L} ||W^{[j]}||_F^2 \tag{25}$$

$$||W^{[j]}||_F^2 = \sum_{i=1}^{n[j-1]} \sum_{k=1}^{n[j]} (W_{ik})^2 \quad \text{(Frobenius norm)} \tag{26}$$

$$dW^{[j]} = \frac{1}{m} dZ^{[j]} A^{[j-1]} + \frac{\lambda}{m} W^{[j]} \tag{27}$$

$$W^{[j]} = W^{[j]} - \alpha dW^{[j]} = W^{[j]} - \frac{\alpha}{m} dZ^{[j]} A^{[j-1]} - \frac{\alpha\lambda}{m} W^{[j]} \tag{28}$$

2.6 Parameter initialization

Sometimes, we observe exploding or vanishing gradients. In order to reduce this effect, we can change the random initialization of the W weights. The choice of the random initialization can be an hyperparameter we tune.

The random expressions are written with Python syntax.

2.6.1 ReLU

$$W^{[j]} = np.random.randn((n[l], n[l-1])) * \sqrt{\frac{2}{n[l-1]}}$$

2.6.2 tanh

$$W^{[j]} = np.random.randn((n[l], n[l-1])) * \sqrt{\frac{1}{n[l-1]}}$$

2.6.3 Other

The following expression is sometimes also used for the random initialization.

$$W^{[j]} = np.random.randn((n[l], n[l-1])) * \sqrt{\frac{2}{n[l-1] + n[l]}}$$

2.7 Feature normalization

Normalizing features can help with the learning.

Suppose $X \in \mathbb{R}^{n_x \times m}$, where m is the number of examples in the dataset and n_x is the number of features. Let $X_k^{(i)}$ be the k-th feature of example i.

$$\mu_k = \frac{1}{m} \sum_{i=1}^{m} X_k^{(i)} \tag{29}$$

$$\sigma_k^2 = \frac{1}{m} \sum_{i=1}^{m} (X_k^{(i)})^2 \tag{30}$$

$$X_k^{(i)} = \frac{X_k^{(i)} - \mu_k}{\sigma_k^2}, \quad \forall i \in \{1, \ldots, m\} \tag{31}$$

3 Other optimization algorithms

3.1 Gradient descent with momentum

Let α and β be two fixed hyperparameters. For each hidden layer i, we initialize $V_{dW^{[i]}} = 0$ and $V_{db^{[i]}} = 0$.

Then, at each epoch we update $V_{dW^{[i]}}$, $V_{db^{[i]}}$, $dW^{[i]}$, and $db^{[i]}$ as follows.

$$V_{dW^{[i]}} = \beta V_{dW^{[i]}} + (1 - \beta) dW^{[i]} \tag{32}$$

$$V_{db^{[i]}} = \beta V_{db^{[i]}} + (1 - \beta) db^{[i]} \tag{33}$$

$$W^{[i]} = W^{[i]} - \alpha V_{dW^{[i]}} \tag{34}$$

$$b^{[i]} = b^{[i]} - \alpha V_{db^{[i]}} \tag{35}$$

3.2 Gradient descent with RMS prop

Let α and β be two fixed hyperparameters. Let $\epsilon = 10^{-8}$. For each hidden layer i, we initialize $S_{dW^{[i]}} = 0$ and $S_{db^{[i]}} = 0$.

Then, at each epoch we update $S_{dW^{[i]}}$, $S_{db^{[i]}}$, $dW^{[i]}$, and $db^{[i]}$ as follows.

$$S_{dW^{[i]}} = \beta S_{dW^{[i]}} + (1-\beta)(dW^{[i]})^2 \tag{36}$$
$$S_{db^{[i]}} = \beta S_{db^{[i]}} + (1-\beta)(db^{[i]})^2 \tag{37}$$
$$W^{[i]} = W^{[i]} - \alpha \frac{dW^{[i]}}{\sqrt{S_{dW^{[i]}}} + \epsilon} \tag{38}$$
$$b^{[i]} = b^{[i]} - \alpha \frac{db^{[i]}}{\sqrt{S_{db^{[i]}}} + \epsilon} \tag{39}$$

3.3 Adam optimization algorithm

Let α, β_1, and β_2 be three fixed hyperparameters. Let $\epsilon = 10^{-8}$. For each hidden layer i, we initialize $V_{dW^{[i]}} = 0$, $V_{db^{[i]}} = 0$, $S_{dW^{[i]}} = 0$, and $S_{db^{[i]}} = 0$.

At each epoch t, we update $V_{dW^{[i]}}$, $V_{db^{[i]}}$, $S_{dW^{[i]}}$, $S_{db^{[i]}}$, $dW^{[i]}$, and $db^{[i]}$ as follows.

$$V_{dW^{[i]}} = \beta_1 V_{dW^{[i]}} + (1-\beta_1) dW^{[i]} \tag{40}$$
$$V_{db^{[i]}} = \beta_1 V_{db^{[i]}} + (1-\beta_1) db^{[i]} \tag{41}$$
$$S_{dW^{[i]}} = \beta_2 S_{dW^{[i]}} + (1-\beta_2)(dW^{[i]})^2 \tag{42}$$
$$S_{db^{[i]}} = \beta_2 S_{db^{[i]}} + (1-\beta_2)(db^{[i]})^2 \tag{43}$$
$$V^{corr}_{dW^{[i]}} = \frac{V_{dW^{[i]}}}{1-\beta_1^t} \tag{44}$$
$$V^{corr}_{db^{[i]}} = \frac{V_{db^{[i]}}}{1-\beta_1^t} \tag{45}$$
$$S^{corr}_{dW^{[i]}} = \frac{S_{dW^{[i]}}}{1-\beta_2^t} \tag{46}$$
$$S^{corr}_{db^{[i]}} = \frac{S_{db^{[i]}}}{1-\beta_2^t} \tag{47}$$
$$W^{[i]} = W^{[i]} - \alpha \frac{V^{corr}_{dW^{[i]}}}{\sqrt{S^{corr}_{dW^{[i]}}} + \epsilon} \tag{48}$$
$$b^{[i]} = b^{[i]} - \alpha \frac{V^{corr}_{db^{[i]}}}{\sqrt{S^{corr}_{db^{[i]}}} + \epsilon} \tag{49}$$

Usually, we set $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Note that ϵ can also be an hyperparameter.

4 Multi-class Classification

When the labels can be more than 0 or 1, we need a different activation function than the typical sigmoid function. Usually, the softmax function is used as the last layer of the neural network.

Let C be the number of possible classes and m be the number of examples in the dataset. The labels $Y \in \{0, 1\}^{C \times m}$ are such that $Y_k^{(i)} = 1$ if example i is labeled with class k.

$$Z^{[j]} = W^{[j]} A^{[j-1]} + b^{[j]} \tag{50}$$

$$A^{[j]} = \frac{e^{Z^{[j]}}}{\sum_{k=1}^{C} e^{Z_k^{[j]}}} \tag{51}$$

The loss function also requires some adaptation.

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{C} y_j \log(\hat{y}_j) \tag{52}$$

$$J(\hat{Y}, Y) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{Y}^{(i)}, Y^{(i)}) \tag{53}$$

5 Improving Results

Poor results can stem from different problems. It is useful to identify what type of poor results we have before trying solutions. Here is a summary.

	Human level error
Avoidable bias	• Train bigger model (more layers, more hidden units, etc.) • Train longer • Better optimization algorithm (momentum, RMSprop, Adam) • Neural Network architecture (activation functions, RNN, CNN, etc.) • Hyperparameters search
	Training set error
Variance	• More data (data augmentation) • Regularization • Neural network architecture • Hyperparameters search
	Development set error

When the algorithm requires more data but we do not have more data on hand, there are still a few solutions we can try.

1. Artificial data synthesis
   For example, we can create audio files with background noise by adding random background noises to our audio tracks. Note that the added noise should be as random as possible to avoid overfitting.

2. Transfert learning
   For example, if there is a training set of 1 000 000 animal images, we can train on those images to learn a lot on image classification. Then, we can learn and tune using our target dataset, 100 images of radiology diagnosis for instances. Sometimes, we add some layers at the already trained neural network for the new information.

6 Multi-task learning

The difference between multi-class classification and multi-task learning is that the outout vector $y \in \{0,1\}^{C \times 1}$ (where C is the number of classes in both cases) only has one non null entry for the multi-class classification while it can have more than one for the multi-task learning problem. One application is the detection of multiple objects in a picture.

Let $Y_j^{(i)}$ be the j-th composant associated to class j for the i-th example in the dataset.

$$\mathcal{L}(Y_j^{(i)}, \hat{Y}_j^{(i)}) = -Y_j^{(i)} \log(\hat{Y}_j^{(i)}) - (1 - Y_j^{(i)}) \log(1 - \hat{Y}_j^{(i)}) \tag{54}$$

$$J(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{C} \mathcal{L}(Y_j^{(i)}, \hat{Y}_j^{(i)}) \tag{55}$$

It might take a lot of time to label examples and some information might be missing from the dataset. This is fine as long as we can clearly identify which information is missing. For example, $y = [0, 1, ?, ?]^T$ means that the first item is not present, the second is present, and we ignore the answer for the last two items. The cost function needs to be adapted in this situation so we only sum over the information we know.

$$J(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{\substack{j \in \{1,\dots,C\} \\ Y_j^{(i)} \in \{0,1\}}} \mathcal{L}(Y_j^{(i)}, \hat{Y}_j^{(i)}) \tag{56}$$