CourseraDeepLearning

April 26, 2018

Contents

%#+header: :file (by-backend (html "tree.svg") (t 'nil)) %#+header: :imagemagick

1 week1

1.1 logistic regression

the output of y in supervised learning problem are either zero or 1

- given $x \in \mathbb{R}^{n_x}$, want $\hat{y} = p(y = 1 \mid x)$,parameters $w \in \mathbb{R}^{n_x}, b \in \setminus$ output $\hat{y} = \sigma(w^t x + b)$,$\sigma(z) = \frac{1}{1+e^{-z}}$. training example:$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want$\hat{y}^{(i)} \approx y^{(i)}$

  loss function measure how good $\hat{y}$ is when the true label is y $\boldsymbol{l}(\hat{y}, y) = -(y\log\hat{y} + (1-y)\log(1-\hat{y}))$ if y = 1, $\boldsymbol{l} = -\log\hat{y}$, want $\hat{y}$ large if y = 0, $\boldsymbol{l} = -\log(1-\hat{y})$,want $\hat{y}$ small

  cost function entire training set measures how well we're doing an entire training set $j(w, b) = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{l}(\hat{y}^{(i)}, y^{(i)})$

1.2 gradient descent

- want to find w, b that minimize $j(w, b)$
- $w := w - \alpha\frac{\partial j(w,b)}{\partial w}$

1.3 logistic regression gradient descent

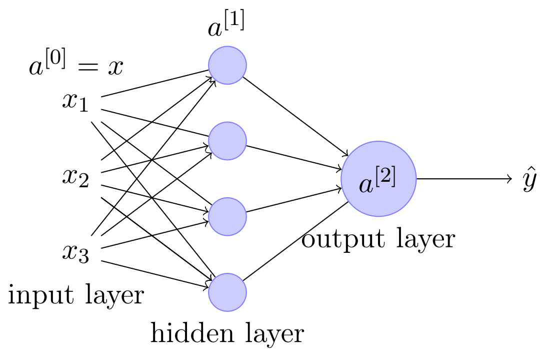
$$\frac{\partial \boldsymbol{l}(a, y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$$
$$\frac{\partial a}{\partial z} = \frac{-e^{-x}}{(1+e^{-x})^2} = a(1-a)$$
$$\frac{\partial \boldsymbol{l}(a, y)}{\partial z} = a(1-a)(-\frac{y}{a} + \frac{1-y}{1-a}) = a - y$$

1.4 vectorization

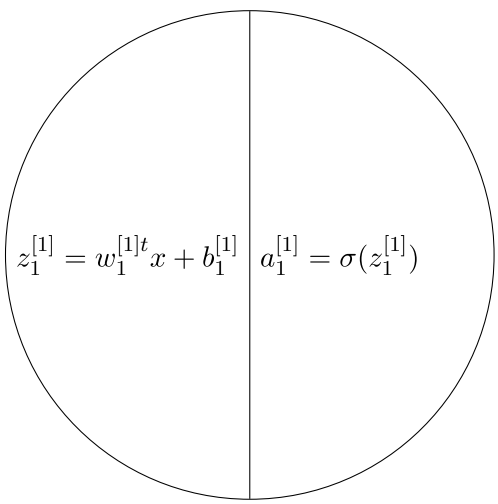
simd single instantion multiple data

2 week2

2.1 neural network representaion



- 2 layer nn



$$
\begin{aligned}
z_1^{[1]} &= w_1^{[1]t} x + b_1^{[1]} \quad a_1^{[1]} = \sigma(z_1^{[1]}) \\
z_2^{[1]} &= w_2^{[1]t} x + b_2^{[1]} \quad a_2^{[1]} = \sigma(z_2^{[1]}) \\
z_3^{[1]} &= w_3^{[1]t} x + b_3^{[1]} \quad a_3^{[1]} = \sigma(z_3^{[1]}) \\
z_4^{[1]} &= w_4^{[1]t} x + b_4^{[1]} \quad a_4^{[1]} = \sigma(z_4^{[1]}) \\
z^{[2]} &= w^{[2]t} a^{[1]} + b^{[2]} \quad a^{[2]} = \sigma(z^{[2]})
\end{aligned}
$$

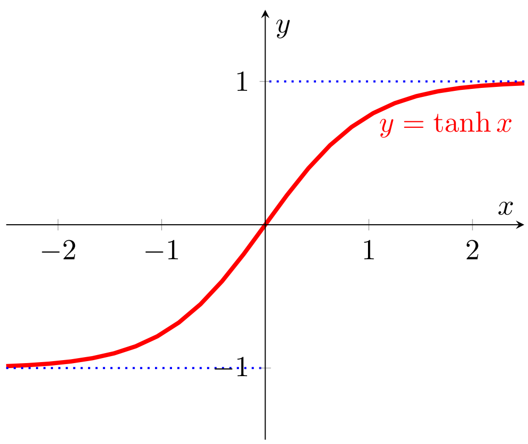
$$\begin{pmatrix} | & | & \cdots & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(n)} \\ | & | & \cdots & | \end{pmatrix}$$

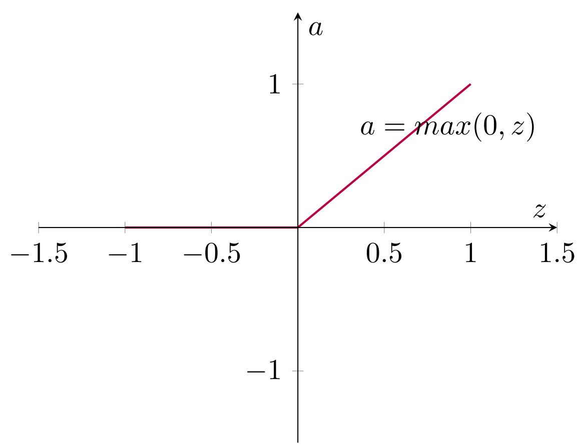
2.2 Activation function

- sigmoid function
- hypobolic tangent function

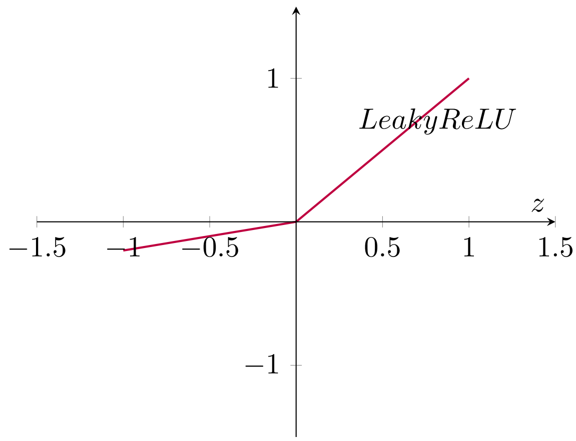
$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - (tanh(z))^2$$



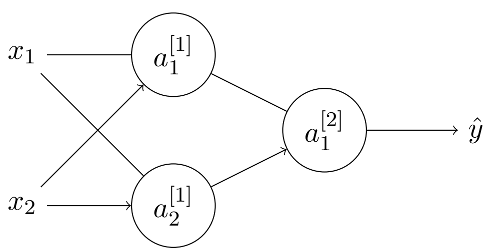


- rectified linear unit



- Leaky ReLU
- sigmoid function: never use except for output
- tanh is better
- ReLU: commonly used

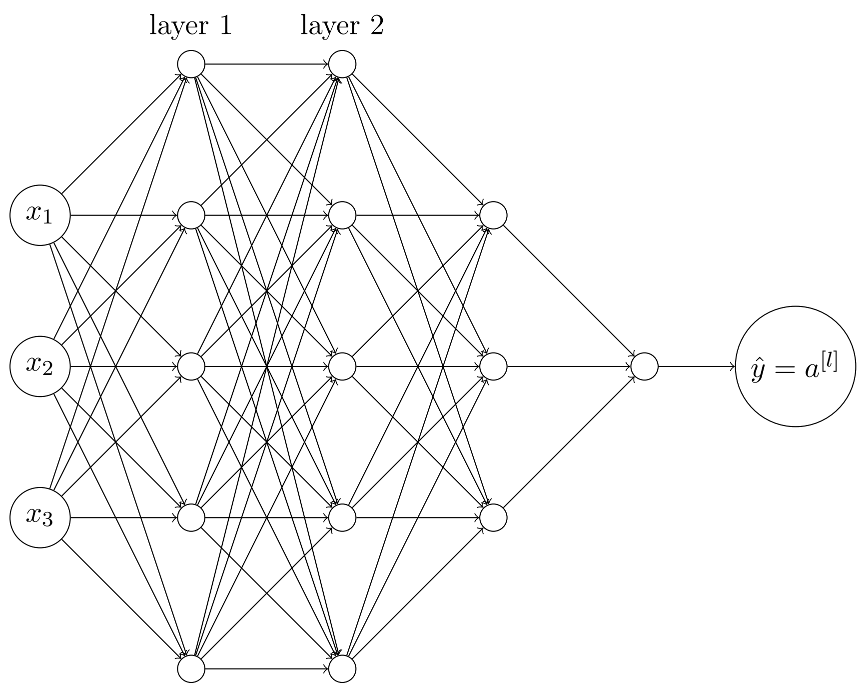
2.3 Random initialization



if w is 0 $a_1^{[1]}$ will be the same as $a_2^{[1]}$

3 Week3

3.1 Deep neural network notation



l is the \$l\$th layer $n^{[l]} = \#units$ in layer l $a^{[l]} = g^{[l]}(z^{[l]})$ is activations in l

3.2 Circuit theory and deep learning

Informally: There are functions you can compute with a "small" L-layer deep neural network that shallower networks require exponentially more hidden units to compute

4 Week4

4.1 Bias and variance

Train set error	1%	15%	15%	1%
Dev set error	11%	16%	30%	0.5%
	high variance	high bias	high bias and variance	low

4.2 Basic recipe for machine learning

$$
\begin{tikzcd}
\begin{array}{c}\text{High bias}\\ \text{training data problem}\end{array} \arrow[rr, bend right, "Y"'] \arrow[d, "N"] & & \begin{array}{c}\text{Bigger network}\\ \text{train longer, NN architercture}\end{array} \arrow[ll, bend right] \\
\begin{array}{c}\text{High variance}\\ \text{dev set problem}\end{array} \arrow[rr, "Y"] \arrow[d, "N"] & & \begin{array}{c}\text{More data}\\ \text{regulation, NN architecture}\end{array} \arrow[u] \\
\text{Done} & &
\end{tikzcd}
$$

4.3 Regularization

L2 regulation–logistic regression $\mathcal{J}(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \dfrac{\lambda}{2m}||w||_2^2$

λ is regulazation parameter

neural network

- $\mathcal{J}(w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) + \dfrac{\lambda}{2m}||w^{[l]}||_F^2$

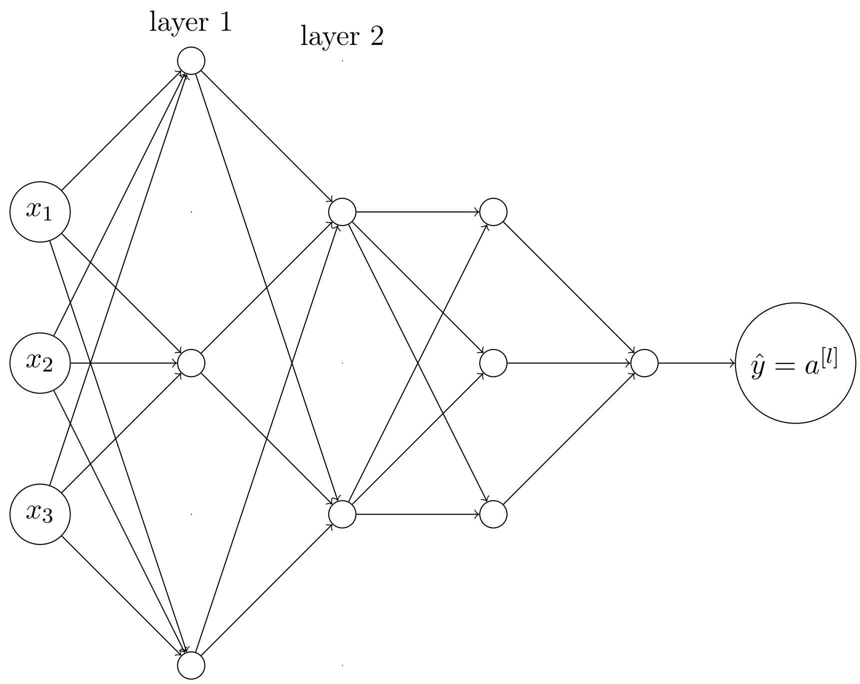
Frobenius norm

- $||w^{[l]}||^2 = \sum_{i=1}^{n^{[l-1]}}\sum_{j=1}^{n^{[l]}}(w_{ij}^{[l]})^2$
- $dw^{[l]} = \cdots + \frac{\lambda}{m}w^{[l]}$

Why regulazation

- If we set $\lambda \to$ big enough, the frobenius norm may tend to approach to 0, which will make some $w^{[l]}$ to be 0 as if hidden layer become just logistic regression, thus overfit may change to just right or high bias.
- If we use *tanh* as activation function, $\lambda \uparrow$, $w^{[l]} \downarrow$,$z^{[l]} = w^{[l]}a^{[l-1]} + b^{[l]} \downarrow$, notice in *tanh*, when $z \to 0$, it tends to be linear function, thus handle the overfitting problem

dropout regulation



implementing dropout

Intuition
- Can't rely on any one feature, so have to spread out weights since they can go away randomly

other methods
- data augmentation i.e. picture : flip, rotate
- early stopping

4.4 Setting up your optimization problem

- normalizing inputs
- vanishing / exploding gradients
- Weight initialization for deep networks

```
W[l] = np.random.rand(shape) * np.sqrt(1 / n[l - 1])
```

- Gradient check for a neural network Take $W^{[1]}, b^{[1]}, \ldots, W^{[L]}, b^{[L]}$ and reshape into a big vector θ $\mathcal{J}(W^{[1]}, b^{[1]}, \ldots, W^{[L]}, b^{[L]}) = \mathcal{J}(\theta)$ Take $dW^{[1]}, db^{[1]}, \ldots, dW^{[L]}, db^{[L]}$ into $d\theta$ Now does $d\theta$ is the gradient of $\mathcal{J}(\theta)$

- for each i $d\theta_{approx}[i] = \frac{\mathcal{J}(\theta_1,\ldots,\theta_i+\epsilon,\ldots)-\mathcal{J}(\theta_1,\ldots,\theta_i+\epsilon,\ldots)}{2\epsilon} \approx \theta[i]$
- Check $\frac{||d\theta_{approx}-d\theta||_2}{||d\theta_{approx}||_2-||d\theta||_2}$
- don't use in training – only in debug