Weekly Report(July.30,2018-Aug.5,2018)

Zhang Yuandi
15826138027@163.com

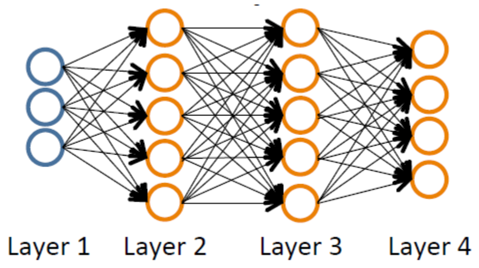
Abstract

Sorry for my absence for two and a half weeks because of numerous assignments in school and social practice. In the last week, I have learned week5, week6 and week7 courses of **Machine Learning** and tried to run the code for **Object Detection**.

1 Work done in this week

1.1 Neural Networks

1.1.1 Cost Function



Let's define a few variables that we will need:

- L = total number of layers in the network
- s$_l$ = number of units (not counting bias unit) in layer 1
- K = number of output units/classes

Recall that the cost function for regularized logistic regression was:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

For neural networks, the cost function is going like this:

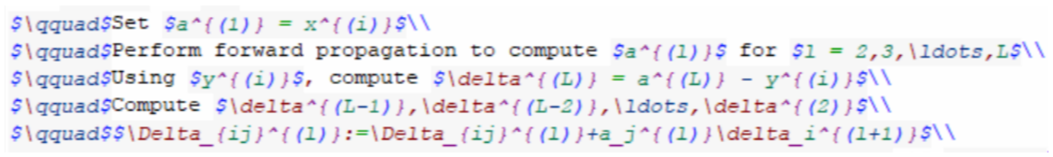
$$h_\Theta(x) \in \mathbb{R}^K \qquad (h_\Theta(x))_i \in i^{th}\text{output}$$

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}[y_k^{(}i)\log((h_\Theta(x^{(i)}))_k)+(1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)]+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}(\Theta_{j,i}^{(l)})^2$$

This is so confusing that I still couldn't fully understand it. How can I remember these complex cost functions and make good use of them?

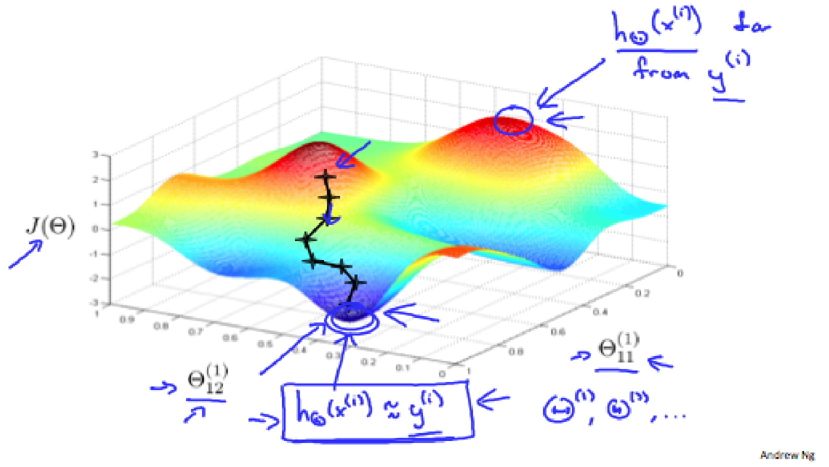
1.1.2 Backpropagation Algorithm

To minimize our neural-network cost function, we use the Backpropagation algorithm:
We have the training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$
Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j)
For $i = 1$ to m
Set $a^{(1)} = x^{(i)}$
Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$
Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$
Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$
$\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$
$D_{ij}^{(1)} := \frac{1}{m}\Delta_{ij}^{(l)} + \lambda\Theta_{ij}^{(l)}$ if $j \neq 0$
$D_{ij}^{(1)} := \frac{1}{m}\Delta_{ij}^{(l)}$ if $j = 0$
$\frac{\partial}{\partial\Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$
Here I have a problem with writing. How to add spaces correctly?


```
$\qquad$Set $a^{(1)} = x^{(i)}$\\
$\qquad$Perform forward propagation to compute $a^{(l)}$ for $l = 2,3,\ldots,L$\\
$\qquad$Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$\\
$\qquad$Compute $\delta^{(L-1)},\delta^{(L-2)},\ldots,\delta^{(2)}$\\
$\qquad$$\Delta_{ij}^{(l)}:=\Delta_{ij}^{(l)}+a_j^{(l)}\delta_i^{(l+1)}$\\
```

As I write in LaTeX, I wanted to add some spaces in the beginning of my lines, but actually it didn't work.
To train a neural network, we can do as follows:

1. Randomly initialize the weights
2. Implement forward propagation to get $h_\Theta(x^{(i)})$ for any $x^{(i)}$
3. Implement the cost function
4. Implement backpropagation to compute partial derivatives
5. Use gradient checking to confirm that your backpropagation works. Then disable gradient checking.
6. Use gradient descent or a built-in optimization function to minimize the cost function with the weights in theta.



This image gives us an intuition of what is happening as we are implementing our neural network.

1.2 Evaluating a Learning Algorithm

Deciding what to try next is an essential part for us. Diagnostics can take time to implement, but doing so can be a very good use of our time.

1.2.1 Evaluating our hypothesis

- Learn parameter θ from training data
- Compute test set error:

$$J_{test}(\theta) = -\frac{1}{m_{test}} \sum_{i=1}^{m_{test}} y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log h_\theta(x_{test}^{(i)})$$

- Misclassification error(0/1 misclassification error):

$$err(h_\Theta(x), y) = \begin{cases} 1 & \text{if } h_\Theta(x) \geq 0.5 \text{ and } y = 0 \text{ or } h_\Theta(x) < 0.5 \text{ and } y = 1 \\ 0 & \text{otherwise} \end{cases}$$

1.2.2 Model Selection

Training error:

$$J_{train(\theta)} = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cross Validation error:

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} (h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)})^2$$

Test error:

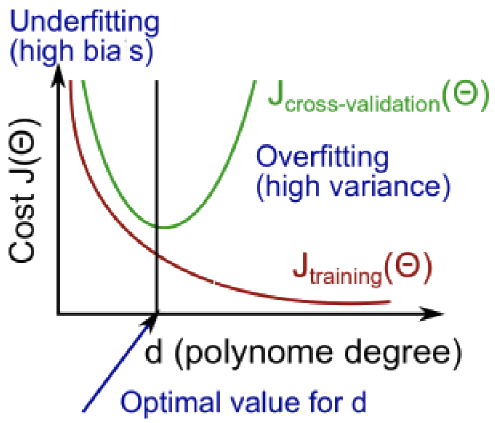
$$J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} (x_{test}^{(i)}) - y_{test}^{(i)})^2$$

In order to choose the model of your hypothesis, we can test each degree of polynomial and look at the error result. We can calculate three separate error values for the three different sets using the following method:

1. Optimize the parameters in Θ using the training set for each polynomial degree.
2. Find the polynomial degree d with the least error using the cross validation set.
3. Estimate the generalization error using the test set with $J_{test}(\Theta^{(d)})$, (d = theta from polynomial with lower error);

1.2.3 Bias vs. Variance

High bias(underfitting): both $J_{train}(\Theta)$ and $J_{CV}(\Theta)$ will be high. Also, $J_{CV}(\Theta) \approx J_{train}(\Theta)$.
High variance(overfitting): $J_{train}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be low and $J_{CV}(\Theta)$ will be much greater than $J_{train}(\Theta)$.
The is summarized in the figure below:



1.2.4 Deciding What to Do Next Revisited

We can do as follows:

- **Getting more training examples:** Fixes high variance
- **Trying smaller sets of features:** Fixes high variance
- **Adding features:** Fixes high bias
- **Adding polynomial features:** Fixes high bias
- **Decreasing λ:** Fixes high bias
- **Increasing λ:** Fixes high variance.

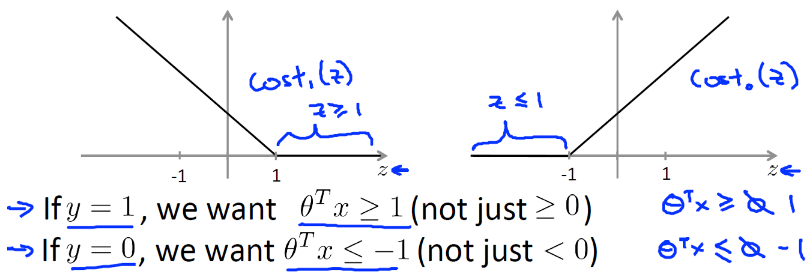
1.3 Support Vector Machines

1.3.1 SVM hypothesis

$$\min_{\theta} C \sum_{i=1}^{m} [y^{(i)} \cos t_1(\theta^T x^{(i)}) + (1 - y^{(i)}) \cos t_0(\theta^T x^{(i)})] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

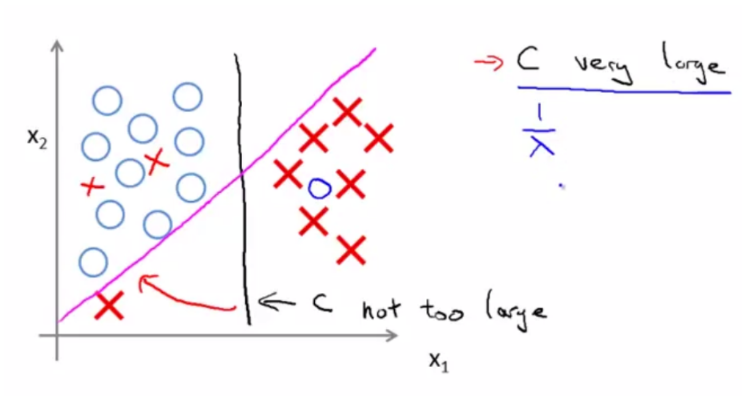
The C parameter is a positive value that controls the penalty for misclassified training examples.

1.3.2 Large Margin Intuition



→ If $\underline{y = 1}$, we want $\underline{\theta^T x \geq 1}$ (not just ≥ 0)
→ If $\underline{y = 0}$, we want $\underline{\theta^T x \leq -1}$ (not just < 0)

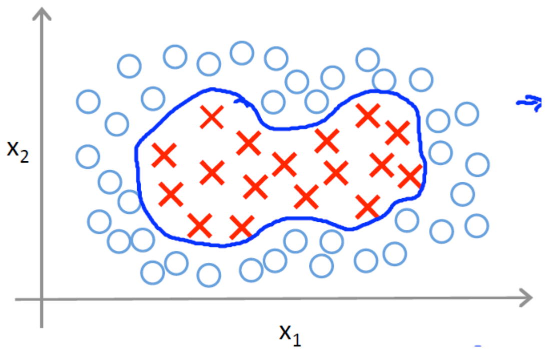
We can understand it intuitively from the image. Also, let's look at this image:



We can see that when C is not too large, we can get the black line. If C is too large, we get the pink line.

1.3.3 Kernels

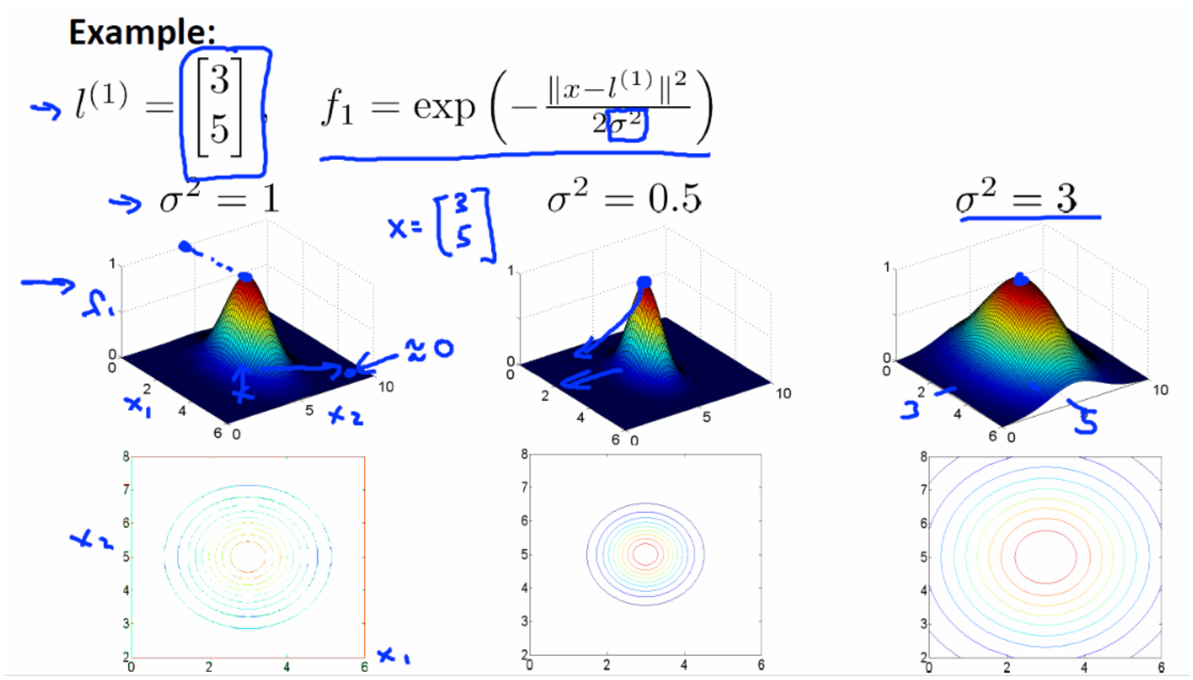
When we're confronted with the case below, what should we do?



Kernels and Similarity

$$f_i = \text{similarity}(x, l^{(i)}) = \exp\left(-\frac{\|x - l^{(i)}\|^2}{2\sigma^2}\right)$$

Here is the example:



It is obvious that the σ parameter determines the decreasing speed. **SVM with Kernels** Given $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^m, y^{(m)})$
Choose $l^{(1)} = x^{(1)}, l^{(2)} = x^{(2)}, \ldots, l^{(m)} = x^{(m)}$
Given example x:

$$f_i = \text{similarity}(x, l^{(i)})$$

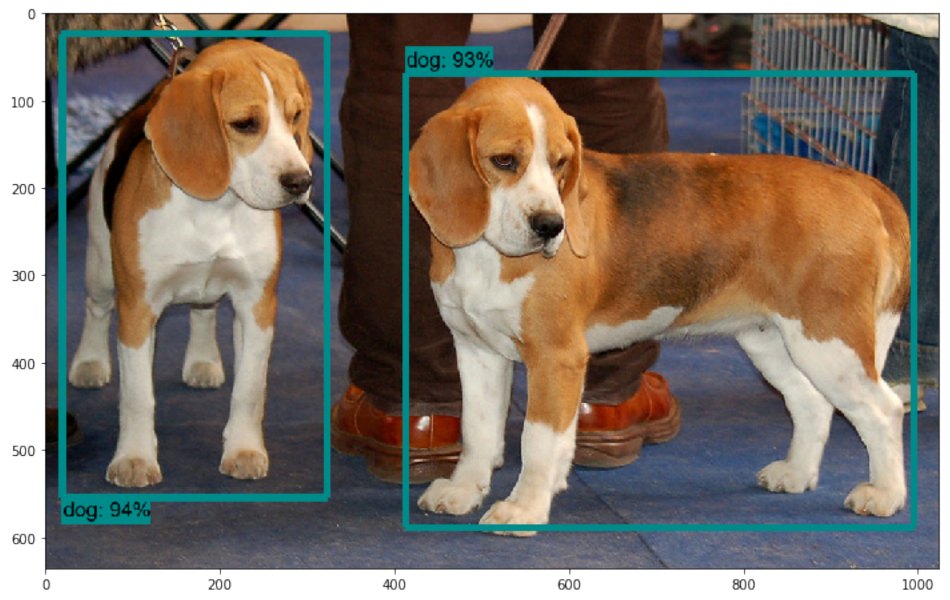
For training example $(x^{(i)}, y^{(i)})$:

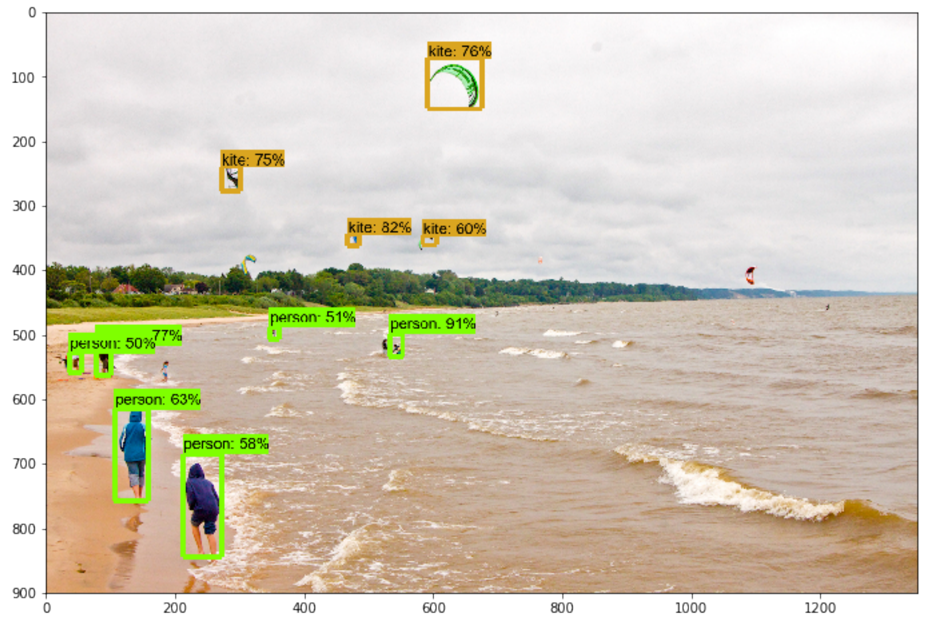
$$f_j^{(i)} = \text{similarity}(x^{(i)}, l^{(j)})$$

And we can get that $C = \frac{1}{\lambda}$. Large C gets lower bias and high variance, And small C gets higher bias and low variance.

1.4 Object Detection

By running the Object Detection Demo, I got the rough outcome:





The matching index is not very high, and I'm figuring how to make it better.

2 Plans for Next Week

1. Learn the week8, week9, week10, week11 course of **Machine Learning**.
2. Do more practices in Detection Networks.