1. RANSAC

I use RANSAC to fit a straight line.

First of all, I randomly elect 2 points to get the corresponding line: \( y = kx + b \).

Secondly, I calculate the divergence between every point to the line: \(|kx + b - y|\). The point whose divergence is within threshold is inliners. Otherwise, as the outliners. Then, denote the number of outliners as the line's error numbers.

Thirdly, repeat the previous steps 800 times and choose the line who own the least error numbers as the proper line.

The results are shown as below:

![RANSAC: \( y = 0.500x + 0.900 \)]

2. AdaBoost

AdaBoost utilize some different weak classifiers to better classify the samples.

First of all, I set same weight weight_D for every sample.
Secondly, use 3 different best weak classifiers and calculate, update the \texttt{weight\_D} for samples and \texttt{weight\_C} for classifiers.

\[
\text{weight\_C} = 0.5 \times \log((1 - \text{error})/\text{error})
\]

\[
\text{weight\_D} = \text{weight\_D} \times \exp(-\text{weight\_C} \times (\text{labels} \times \text{best\_weak\_classifier['result']}))
\]

To get 3 best weak classifier:

1. randomly construct 1 weak classifier with \texttt{dimension}, \texttt{positive\_side}, \texttt{boundary}
2. according to \texttt{weight\_D}, calculate the corresponding error
3. repeat step 1 and 2 100 times and get the \texttt{best\_weak\_classifier} who own the least error

Construct \texttt{adaClassifier} with the 3 \texttt{best\_weak\_classifier} and corresponding \texttt{weight\_C}.

Thirdly, get the \texttt{ada\_result} by classify the samples with \texttt{adaClassifier} and print out:

Results of \texttt{ada\_classifier}: [x, y, label, evaluation\_result] [[80. 144. 1. 1.], [93. 232. 1. 1.], [136. 275. -1. -1.], [147. 131. -1. -1.], [159. 69. 1. 1.], [214. 31. 1. 1.], [214. 152. -1. -1.], [257. 83. 1. 1.], [307. 62. -1. -1.], [307. 231. -1. -1.]] Accuracy: 100.0%

Last, plot the results. The results are shown as below:

3. Math regarding PCA

\texttt{Pf}:
From the condition presented, we are informed that the variance of data projections to $\alpha_2$, as an unit-vector among all the orientations orthogonal to $\alpha_1$, is the largest one:

$$\alpha_2 = \arg\max (\alpha^T C \alpha) \text{ s. t. } \alpha^T \alpha = 1, \alpha^T \alpha_1 = 0$$  \hspace{1cm} (1)

According to Lagrangian multiplier method, we are supposed to maximize $F$:

$$F = \alpha^T C \alpha - \lambda_1 (\alpha^T \alpha - 1) - \lambda_2 \alpha^T \alpha_1$$  \hspace{1cm} (2)

Calculating the first derivative of $F$:

$$\frac{d(F)}{d(\alpha)} = 2C\alpha - 2\lambda_1 \alpha - \lambda_2 \alpha_1 = 0$$  \hspace{1cm} (3)

Left multiplying $\alpha_1^T$:

$$2\alpha_1^T C \alpha - 2\lambda_1 \alpha_1^T \alpha - \lambda_2 \alpha_1^T \alpha_1 = 0$$  \hspace{1cm} (4)

Due to $\alpha_1^T \alpha_1 = 1, \alpha_1^T \alpha = (\alpha^T \alpha_1)^T = 0$:

$$2\alpha_1^T C \alpha - \lambda_2 = 0$$  \hspace{1cm} (5)

Given that the corresponding eigen value of $\alpha_1$, a eigen vector of covariance matrix $C$, is $\lambda_1$:

$$C\alpha_1 = \lambda_1 \alpha_1$$  \hspace{1cm} (6)

$$C^T = C$$  \hspace{1cm} (7)

Combining equation (5),(6),(7):

$$2\alpha_1^T C \alpha - \lambda_2$$
$$= 2(C^T \alpha_1)^T \alpha - \lambda_2$$
$$= 2(C \alpha_1)^T \alpha - \lambda_2$$
$$= 2(\lambda_1 \alpha_1)^T \alpha - \lambda_2$$
$$= 2\lambda_1 \alpha_1^T \alpha - \lambda_2$$
$$= 0$$

Since $\alpha_1^T \alpha = 0$:

$$\lambda_2 = 0$$  \hspace{1cm} (8)

Therefore, the equality (3) as:

$$C\alpha = \lambda_1 \alpha$$  \hspace{1cm} (9)

Hence, $\alpha$ is the eigen vector of the covariance matrix $C$:  

$$\arg\max (\text{var}(\alpha^T X))$$
$$= \arg\max (\alpha^T C \alpha)$$
$$= \arg\max (\alpha^T \lambda \alpha)$$
$$= \arg\max (\lambda)$$
Given that the variance of data projections to $\alpha_3$ is the largest one among all the orientations orthogonal to $\alpha_1$, $\alpha_2$ is the largest eigen-value of vectors orthogonal to $\alpha_1$. According to the condition that is $\alpha_1$ the eigen-vector associated with the largest eigen-value of $C$. Meanwhile, all eigen-vector is orthogonal to each other. Thereby, $\alpha_2$ actually is the eigen-vector of $C$ associated to $C$'s second largest eigen-value.

QED.

4. Math regarding Logistic Regression

Pf:

First of all, let’s demonstrate the equality (1) regarding sigmoid function $f(x) = \frac{1}{1+e^{-x}}$:

$$f(x)' = f(x) (1 - f(x))$$

$$\therefore f(x) (1 - f(x)) = \frac{e^{-x}}{(1 + e^{-x})^2}$$

$$f(x)' = \frac{e^{-x}}{(1 + e^{-x})^2}$$

$$\therefore f(x)' = f(x) (1 - f(x))$$

In addition, according to the definition of cost function, we know that $h_\theta(x_i)$ is the possibility of a sample as true, and $h_\theta(x_i) = f(\theta^T x_i)$. Due to the independence of each sample, for each sample:

$$\nabla_\theta (-(y_i \log(h_\theta(x_i)) + (1 - y_i) \log(1 - h_\theta(x_i))))$$

$$= -(y_i \frac{d\theta}{h_\theta(x_i)} + (1 - y_i) \frac{-d\theta}{1 - h_\theta(x_i)})$$

$$= -x_i (y_i \frac{d\theta^T x_i}{h_\theta(x_i)} + (1 - y_i) \frac{1 - h_\theta(x_i)}{1 - h_\theta(x_i)})$$

$$= -x_i (y_i \frac{d\theta^T x_i}{f(\theta^T x_i)} + (1 - y_i) \frac{1 - f(\theta^T x_i)}{1 - f(\theta^T x_i)})$$

Let $a = \theta^T x_i$, equality (2) will be transformed:

$$-x_i (y_i \frac{d\theta^T x_i}{f(a)} + (1 - y_i) \frac{-d\theta^T x_i}{1 - f(a)})$$

$$= -x_i (y_i \frac{da}{f(a)} + (1 - y_i) \frac{1 - f(a)}{1 - f(a)})$$

Applying the equality (1) to equality (3):
\[ -x_i \left( y_i \frac{df(a)}{da} \frac{df(a)}{f(a)} + (1 - y_i) \frac{df(a)}{1 - f(a)} \right) \]
\[ = -x_i \left( y_i (1 - f(a)) + (1 - y_i) - f(a)(1 - f(a)) \right) \]
\[ = x_i (f(a) - y_i) \]
\[ = x_i (f(\theta^T x_i) - y_i) \]
\[ = x_i (h_\theta(x_i) - y_i) \]

Due to the independence of all samples, sum up their gradients:

\[
\nabla_\theta J(\theta) = \sum_{i=1}^{m} x_i (h_\theta(x_i) - y_i)
\]

QED.

5. Recognize and Locate 2 kinds of Products with YoloV2

5.0 Environment

- Ubuntu: 16.04
- GPU: GTX 1080 Titan

5.1 Collect and Label Images

- we collect samples at Tianmao supermarket, Kuaike vendor, Xianbaohui supermarket, Wal-Mart using smartphone's camera. The total number of collected samples is 102.
- we use tool LabelImg to label the images:
5.2 Augment data

We augment raw data in four different ways in order to improve the generalization ability of the network. We get the augmented data set consisted of 306 images by 3 ways:

- We randomly rotate the images from 10 to 20 degrees
- We implement Gaussian blur on these images
- We randomly crop the images and resize to original size

<table>
<thead>
<tr>
<th>original</th>
<th>rotate</th>
<th>Gaussian blur</th>
<th>crop</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original Image" /></td>
<td><img src="image2" alt="Rotated Image" /></td>
<td><img src="image3" alt="Gaussian Blurred Image" /></td>
<td><img src="image4" alt="Cropped Image" /></td>
</tr>
</tbody>
</table>

5.3 Deploy Darknet

1. download the source code from darknet git repository:

   https://github.com/pjreddie/darknet.git

2. Change `Makefile` in the base directory

   ```
   GPU=1
   CUDNN=1
   OPENCV=1
   ```

3. Compile C code

5.4 YoloV2 Fine-Tuning

1. Download the pretrained convolution weights:


2. Change the configure:
3. Set the training and validation file
   - training samples: 320
   - validation samples: 88

   randomly choose 320 indexes out of 408 as the training samples' indexes, and save name as train.txt. Others as the test.txt

4. Train

```
./darknet detector train cfg/voc.data cfg/yolov2-voc.cfg darknet19_448.conv.23
```

5.5 Save

I use cv2 to save the corresponding predicted images as video.

5.6 Result

The AP in validation data set is 97.63%, and the tested video is attached in the folder.