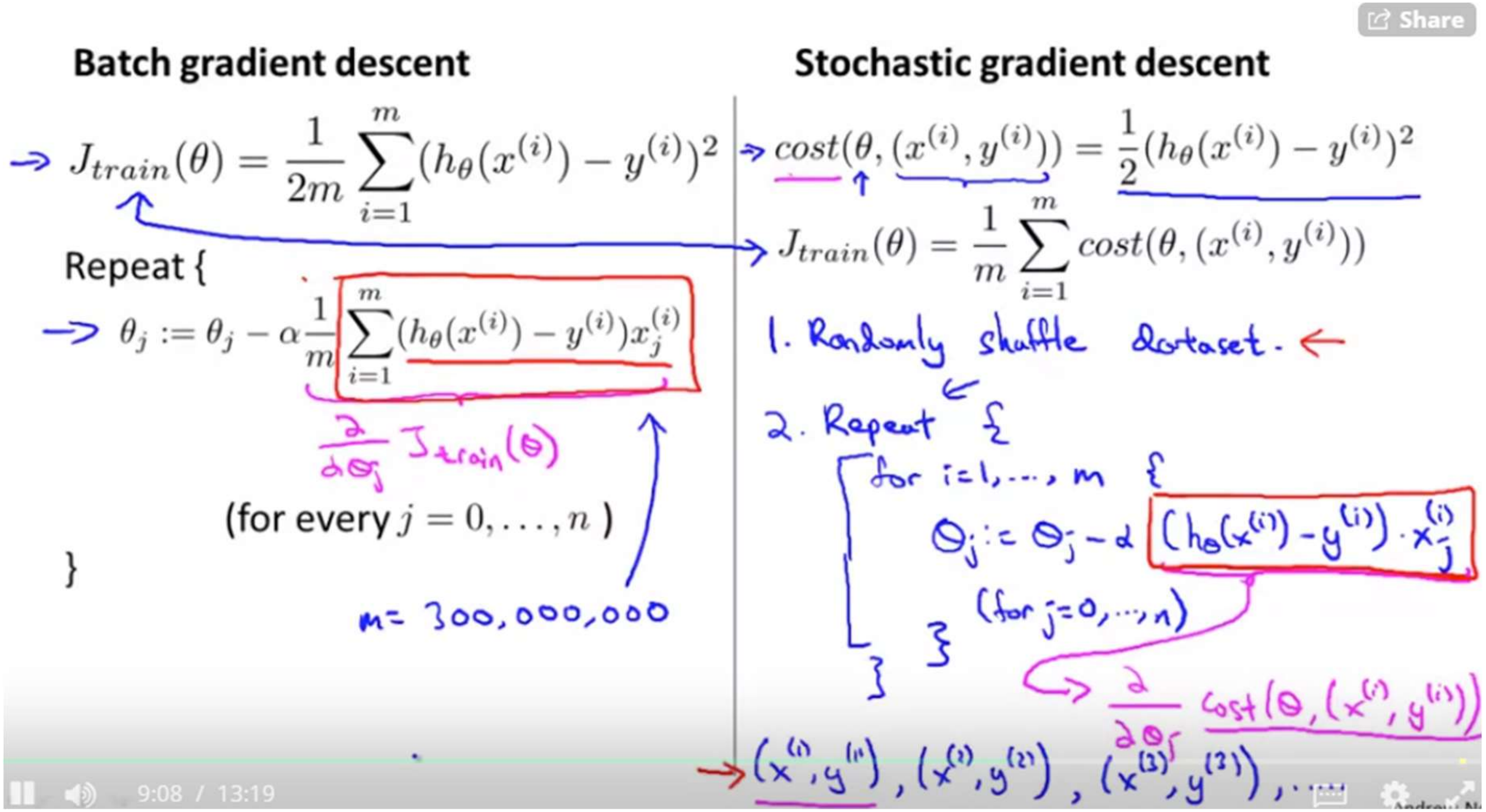
Machine Learning Part VI

Gradient Descent with Large Datasets

Stochastic Gradient Descent

- updating parameter after each training example



Stochastic gradient descent

1. Randomly shuffle (reorder) training examples

2. Repeat { 1-10x

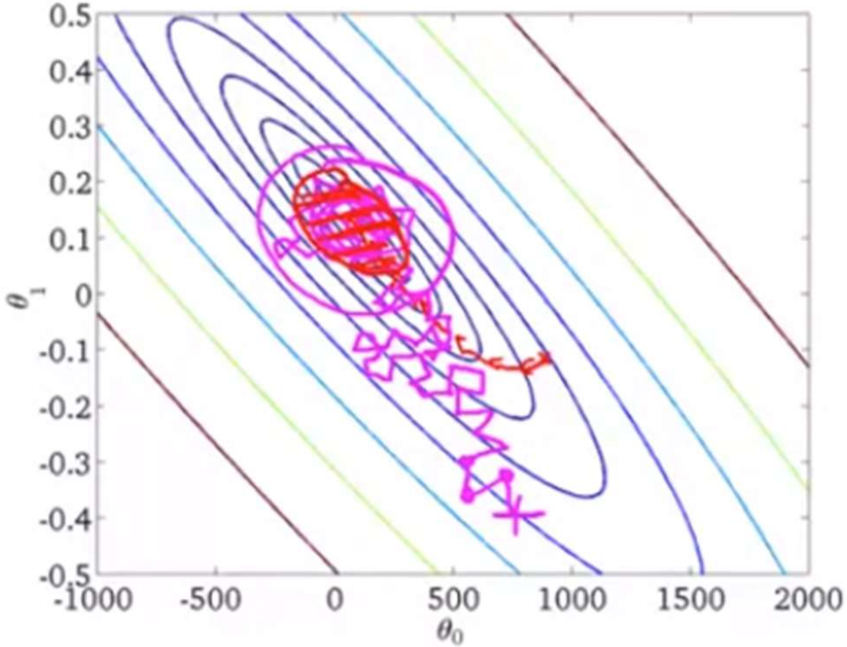
   for $i := 1, \ldots, m$ {

   $$\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

   (for every $j = 0, \ldots, n$)

   }

}



Mini-batch gradient descent

→ Batch gradient descent: Use all m examples in each iteration

→ Stochastic gradient descent: Use 1 example in each iteration

Mini-batch gradient descent: Use b examples in each iteration

b = mini-batch size. $b = 10$. 2 - 100

Get $\boxed{b = 10}$ examples $(x^{(i)}, y^{(i)}), \ldots (x^{(i+9)}, y^{(i+9)})$

$$\theta_j := \theta_j - \alpha \frac{1}{\boxed{10}} \sum_{k=i}^{\boxed{i+9}} \left(h_\theta(x^{(k)}) - y^{(k)}\right) \cdot x_j^{(k)}.$$

$i := i + 10;$

Pause

Mini-batch gradient descent

b examples

1 example

Vectorization

Say $\boxed{b = 10}$, $m = \boxed{1000.}$

Repeat {

for $i = 1, 11, 21, 31, \ldots, \boxed{991}$ {

$$\theta_j := \theta_j - \alpha \boxed{\frac{1}{\boxed{10}} \sum_{k=i}^{\boxed{i+9}} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}}$$

(for every $j = 0, \ldots, n$)

}

}

m = 300,000,000

b = 10

Using 1000 instead of all m training examples

Checking for convergence

- Batch gradient descent:
  - Plot $J_{train}(\theta)$ as a function of the number of iterations of gradient descent.
  - $\boxed{J_{train}(\theta)} = \frac{1}{2m}\boxed{\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2}$

M = 300, 000, 000

- Stochastic gradient descent:
  - $cost(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$
  - During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

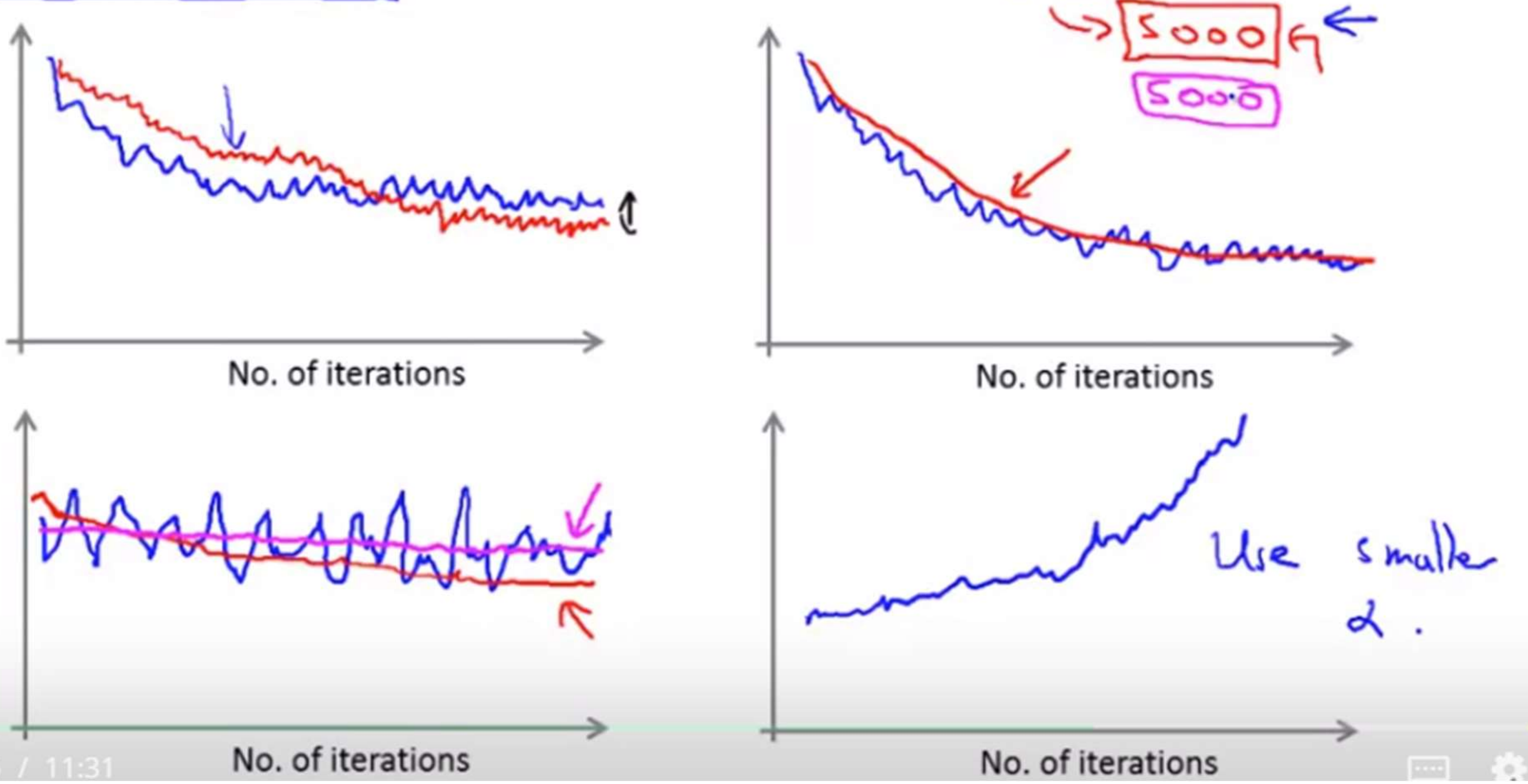
$(x^{(i)}, y^{(i)})$, $(x^{(i+1)}, y^{(i+1)})$, ...

- Every 1000 iterations (say), plot $cost(\theta, (x^{(i)}, y^{(i)}))$ averaged over the last 1000 examples processed by algorithm.

use

Checking for convergence

Plot $cost(\theta,(x^{(i)},y^{(i)}))$, averaged over the last 1000 (say) examples



Stochastic gradient descent

$$cost(\theta,(x^{(i)},y^{(i)})) = \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$$

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

1. Randomly shuffle dataset.
2. Repeat {

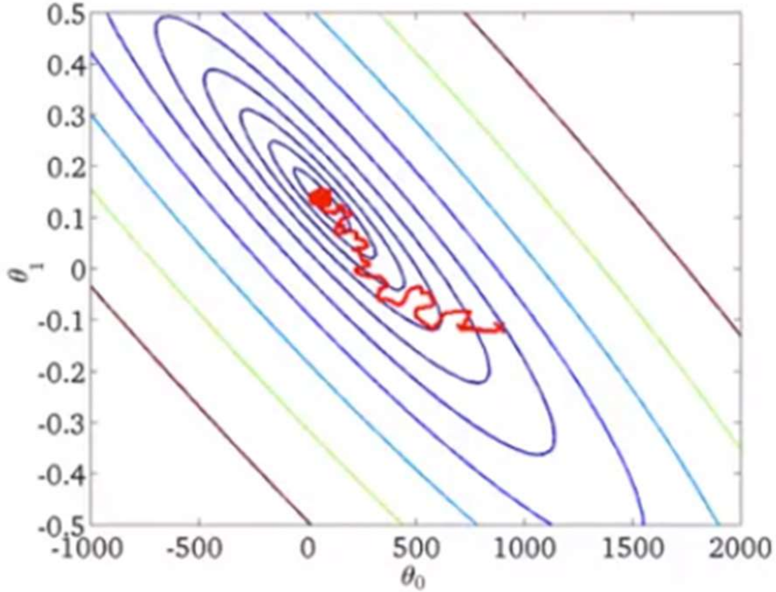
    for $i := 1, \ldots, m$ {

    $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$

    (for $j = 0, \ldots, n$)

    }

}



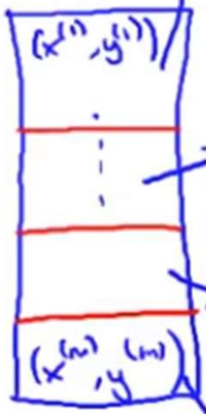
Learning rate α is typically held constant. Can slowly decrease α over time if we want θ to converge. (E.g. $\alpha = \frac{\texttt{const1}}{\boxed{\texttt{iterationNumber}} + \texttt{const2}}$) $\alpha \to 0$

Map-reduce

$\boxed{m = 400}$ $m = 400{,}000{,}000$

Map-reduce

Batch gradient descent: $\theta_j := \theta_j - \alpha \frac{1}{400} \sum_{i=1}^{\boxed{400}} (h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$



Machine 1: Use $(x^{(1)}, y^{(1)}), \ldots, (x^{(100)}, y^{(100)})$.

$temp_j^{(1)} = \sum_{i=1}^{100} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 2: Use $(x^{(101)}, y^{(101)}), \ldots, (x^{(200)}, y^{(200)})$.

$temp_j^{(2)} = \sum_{i=101}^{200} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 3: Use $(x^{(201)}, y^{(201)}), \ldots, (x^{(300)}, y^{(300)})$.

$temp_j^{(3)} = \sum_{i=201}^{300} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

Machine 4: Use $(x^{(301)}, y^{(301)}), \ldots, (x^{(400)}, y^{(400)})$.

$temp_j^{(4)} = \sum_{i=301}^{400} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$

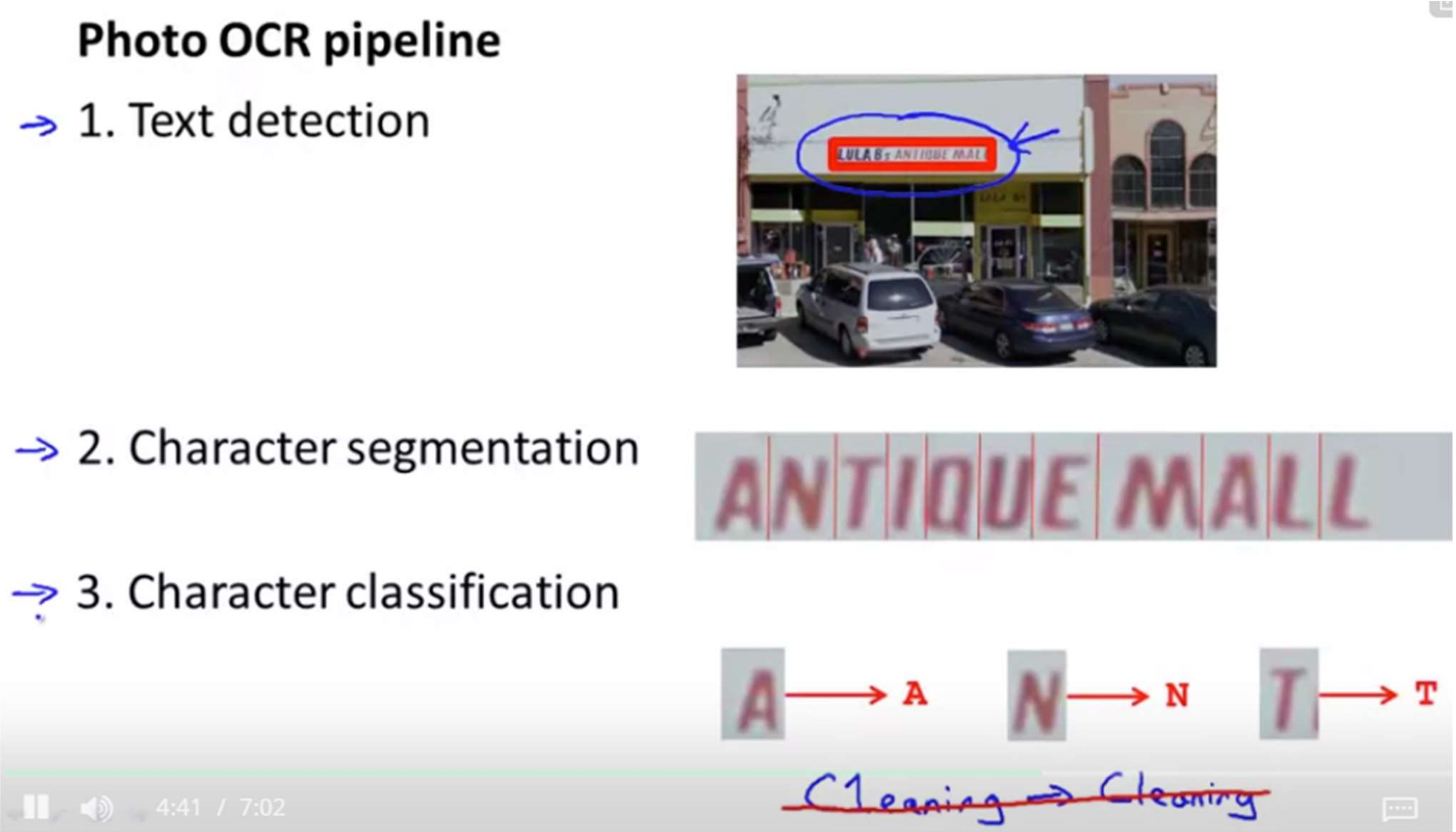
Combine:

$$\theta_j := \theta_j - \alpha \frac{1}{400} (temp_j^{(1)} + temp_j^{(2)} + temp_j^{(3)} + temp_j^{(4)})$$

$(j = 0, \ldots, n)$

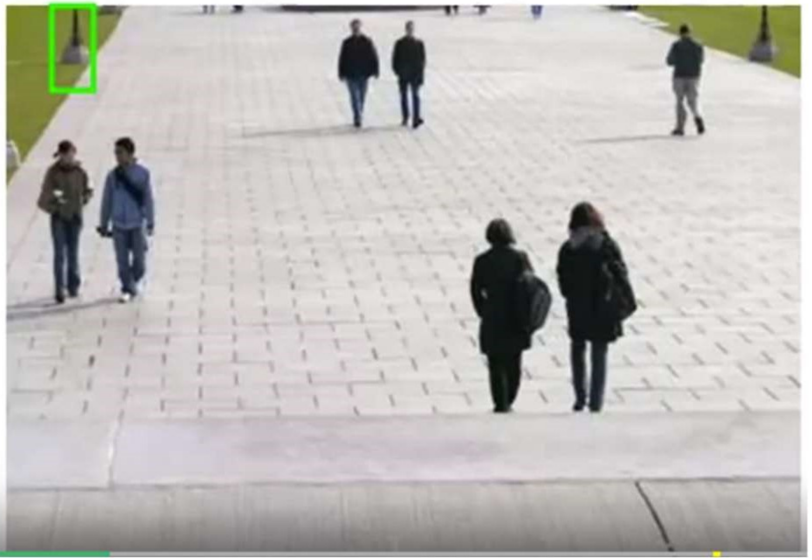
Case study: Photo OCR
Optical character recognition

Machine Learning Pipeline

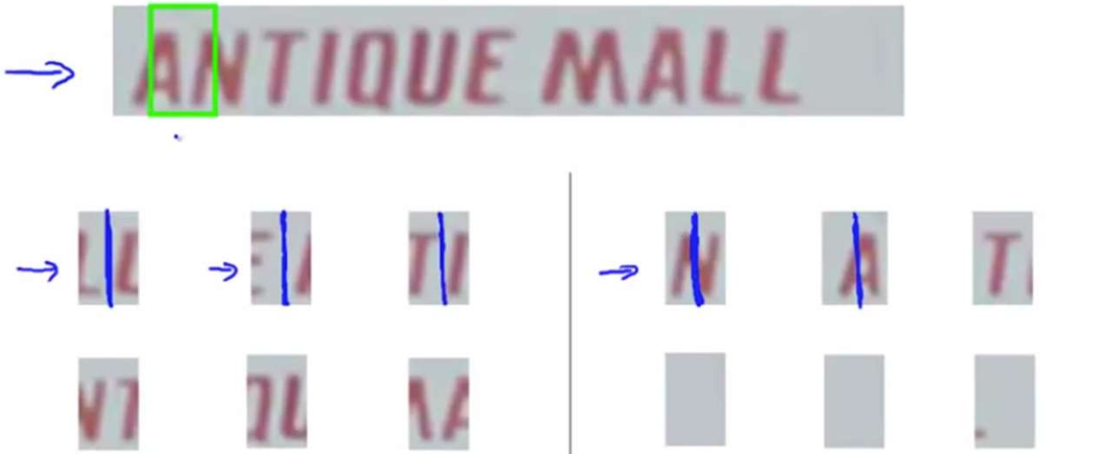


Sliding window detection

step-size /stride



1D Sliding window for character segmentation

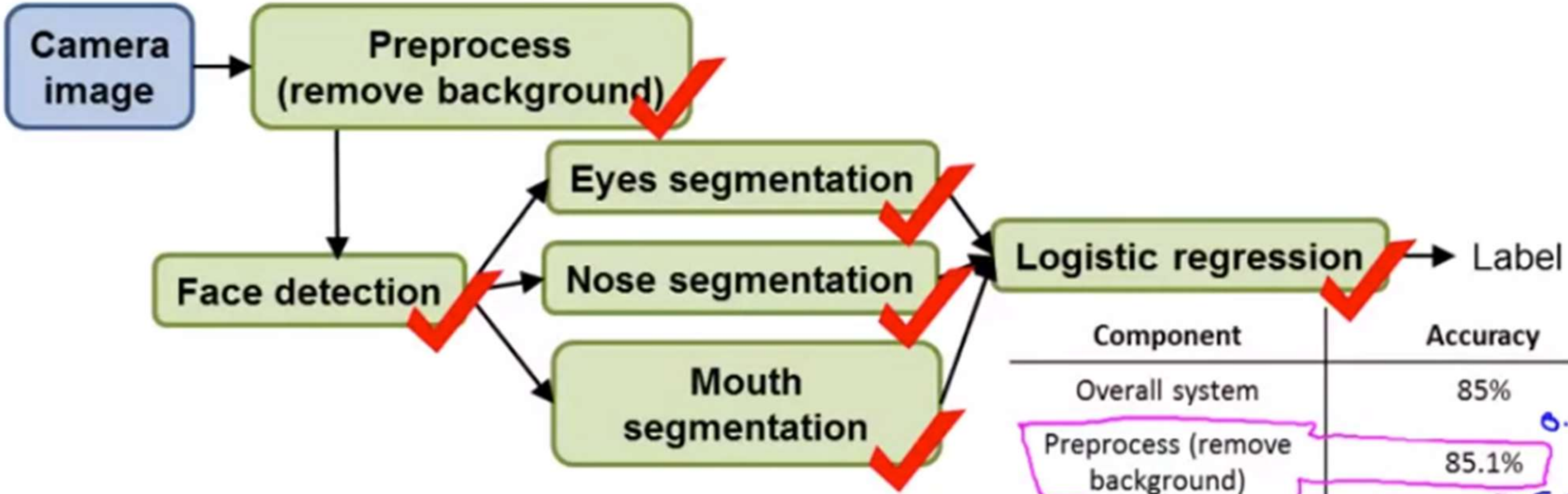


Positive examples $(y = 1)$ Negative examples $(y = 0)$

Discussion on getting more data

1. Make sure you have a low bias classifier before expending the effort. (Plot learning curves). E.g. keep increasing the number of features/number of hidden units in neural network until you have a low bias classifier.
2. "How much work would it be to get 10x as much data as we currently have?"
   - Artificial data synthesis
   - Collect/label it yourself
   - "Crowd source" (E.g. Amazon Mechanical Turk)

Another ceiling analysis example



Component	Accuracy	
Overall system	85%	
Preprocess (remove background)	85.1%	0.1%
Face detection	91%	5.9%
Eyes segmentation	95%	4%
Nose segmentation	96%	1%
Mouth segmentation	97%	1%
Logistic regression	100%	3%