Weekly Report(Aug.6,2018-Aug.12,2018)

Zhang Yuandi
15826138027@163.com

Abstract

In this week, I have finished **Machine learning** on coursera, but haven't finished those exercises yet. Besides, I'm making an endeavour to train the the object detection network **SSD** but failed for numerous inexplicable problems, therefore I didn't write them down this week.

1 Work done in this week

1.1 Clustering

1.1.1 K$-$means algorithm

For unsupervised learning, how can we figure out the inner structure of those data? We can do clustering using K$-$means algorithm:
Input:

- K(number of clusters)
- Training set $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$

$x^{(i)} \in \mathbb{R}^n$(drop $x_0 = 1$ convention)
Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$
Repeat{

$i = 1$ to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for $k = 1$ to K

μ_k := average (mean) of points assigned to cluster k

}

1.1.2 Optimization objective

Cost function:

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - \mu_{c(i)}\|^2$$

Our goal is to minimize the cost function. As we know, the first loop minimizes what $c^{(i)}$ costs, and the second loop minimizes what μ_i costs. Therefore, the cost decreases every iteration. Otherwise, there must be some errors.

1.1.3 Random initialization

To avoid local minimum, we always run K$-$means algorithm for many times, and do random initialization every time.
For $i = 1$ to 100 {

Randomly initialize K$-$means.

Run K$-$means. Get $c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K$.

Compute cost function(distortion) $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$.

}
Pick clustering that gave lowest cost $J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$.

1.2 Dimensionality Reduction

PCA is a common algorithm for dimensionality reduction. Reduce data from $n-$dimensions to $k-$dimensions
Compute "covariance matrix":

$$\sum = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T$$

Compute "eigenvectors" of matrix $\sum$:

$$[U, S, V] = \text{svd(Sigma);}$$

We get:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

After mean normalization and optionally feature scaling:

$$\text{Sigma} = \frac{1}{m} \sum_{i=1}^{m} (x^{(i)})(x^{(i)})^T$$

[U, S, V] = svd(Sigma);
Ureduce = U(:, 1:k);
z = Ureduce'*x;
Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01$$

So "99%" of variance is retained.

1.3 Anomaly detection

1. Choose features x_i that might be indicative of anomalous examples.
2. Fit parameters $\mu_1, \dots, \mu_n, \sigma_1^2, \dots, \sigma_n^2$

$$\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^{(i)}$$

$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

3. Given new example x, compute $p(x)$:

$$p(x) = \prod_{j=1}^{n} p(x_j; \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2})$$

Anomaly if $p(x) < \varepsilon$.

1.4 Large scale machine learning

1.4.1 Stochastic gradient descent

To see whether a large set of data is valuable, we can draw the learning curve. If we has to use a large set of data, stochastic gradient descent is a good choice. We define cost function:

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

1. Randomly shuffle dataset.
2. Repeat{
   for i = 1:m{
   $\theta_j := \theta_j - \alpha(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$
   (for j = 0:n)
   }
   }

1.4.2 Mini–batch gradient descent

Say $b = 10, m = 1000$.
Repeat{

for $i = 1, 11, 21, 31, \dots, 991${

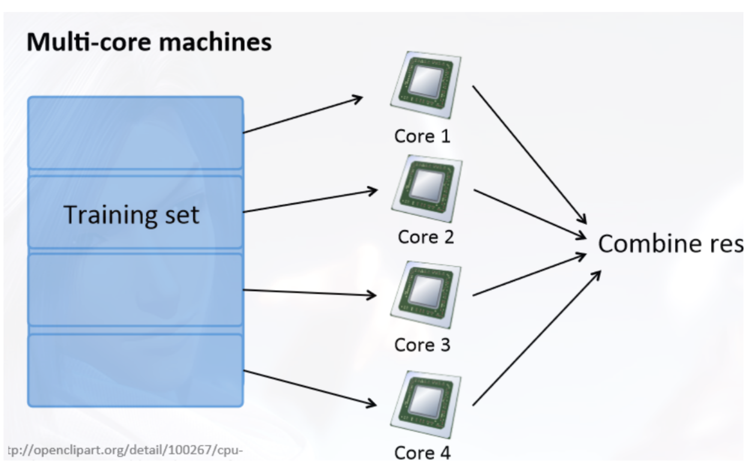
$$\theta_j := \theta_j - \alpha\frac{1}{10}\sum_{k=i}^{i+9}(h_\theta(x^{(k)}) - y^{(k)})x_j^{(k)}$$

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

}
}

1.4.3 Map−reduce and data parallelism

We can assign some mini dataset to a few computers, then combine those results.



After finishing it, I still can't fully understand some algorithms and even forgot some contents learned earlier, which I think is not very efficient. Thus, I plan to take one or two weeks to review it and learn deeper about it. Are there any other useful references for machine learning?

2 Plans for Next Week

1. Review **Machine Learning**.

2. Finish my mathematical modeling assignment.
3. Learn Lecture1, Lecture2, Lecture3 of **CS231n**.
4. Keep on training **SSD**, and train other detection networks if possible.