Advanced Machine Learning and Deep Learning

Dr. Mohammad Pourhomayoun
Assistant Professor
Computer Science Department
California State University, Los Angeles

Working with Large-Scale Data

Why Big Data?

- Why Big Data is an important topic these days?

- Because now we have:

1. **New Sources** of Data that did not exist before.
2. **New Capabilities** to acquire and store data.
3. **New Techniques** to process data.

Large-Scale Machine Learning

- One of the main reasons that machine learning algorithms achieve much better results recently compared to say 5-10 years ago is **the massive datasets that we have for training now**.

- The cost of analyzing data, extracting useful knowledge from it, and learn from it is the **new big bottleneck**.

Large-Scale Machine Learning

- Learning from large datasets is **very challenging**. We need to handle computational complexity!

- **Example**: Suppose that we want to train a linear (or logistic) regression using gradient descent algorithm on $\mathbf{m=100\ Million}$ samples. We need to calculate a summation over 100 Million samples in EACH iteration!

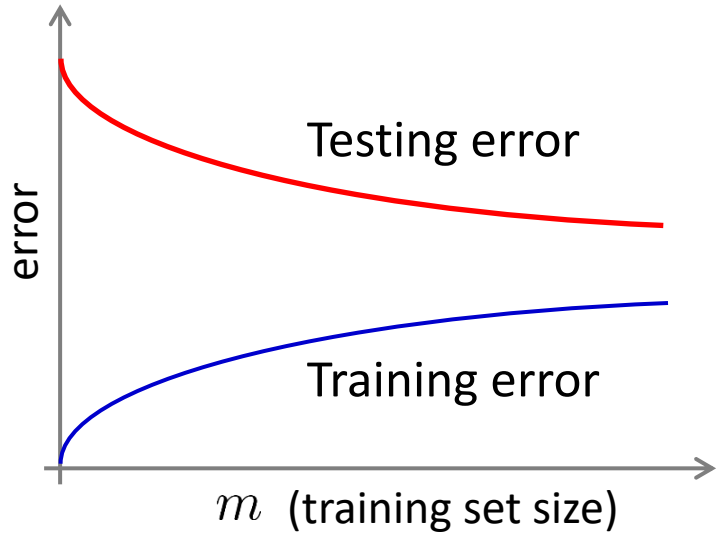
$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\, x_j^{(i)}$$

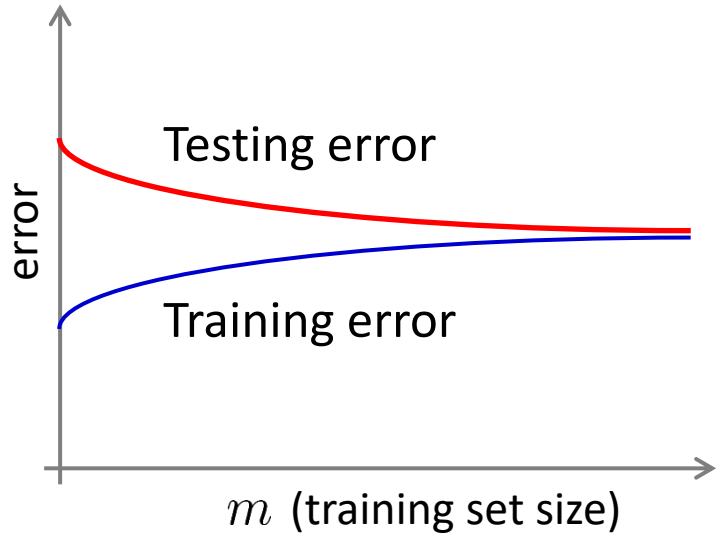
Large-Scale Machine Learning

- There are several approaches to handle **learning from Big Data**:
  1. Modifying the learning algorithms.
     - E.g. Stochastic Gradient Descent (covered in CS4662)
     - E.g. Mini-Batch Gradient Descent (covered in CS4662)
  2. Parallel Computing and MapReduce (covered in CS4661).
  3. Dimensionality Reduction (CS4662).
  4. Sampling the big dataset and only using the samples (Simplest, but not the best approach).

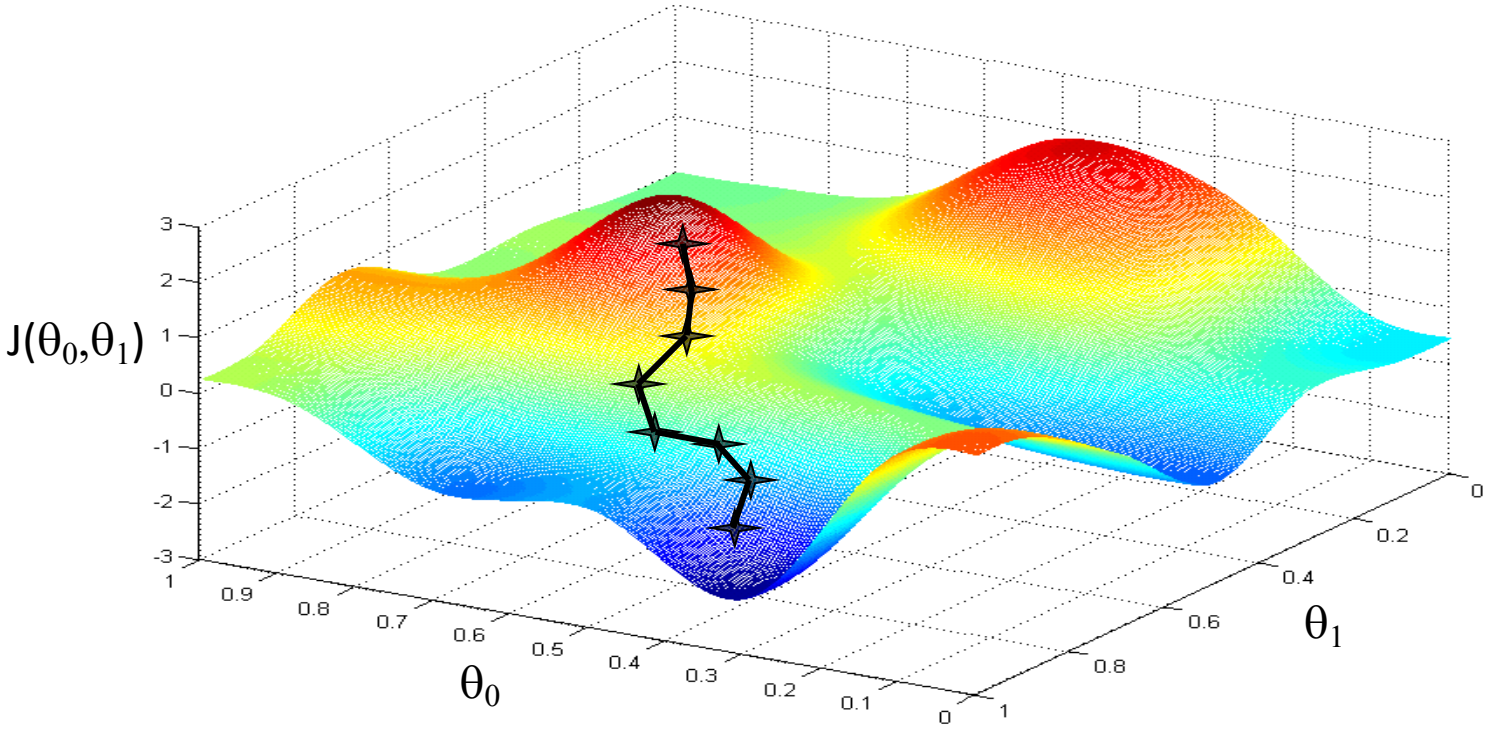
Sampling the Big Data

- **Sampling the Big Data**: Simplest (but not the best) approach.
- If we want to sample the training dataset and only use those samples in training, how many samples should be selected?





Stochastic Gradient Descent for Big Data



Training Set: $\{(x^{(1)},y^{(1)}), (x^{(2)},y^{(2)}), \ldots, (x^{(m)},y^{(m)})\}$

Updates In Each Step: $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\, x_j^{(i)}$

Reference: Andrew Ng, Machine Learning, Stanford University.

Large-Scale Machine Learning

- Suppose that we want to train a linear (or logistic) regression using gradient descent algorithm on **m=100 Million** samples. We need to calculate a summation over 100 Million samples in EACH iteration!

Training Set: $\{(x^{(1)},y^{(1)}), (x^{(2)},y^{(2)}), \ldots, (x^{(100,000,000)},y^{(100,000,000)})\}$

Updates In Each Step: $\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\boldsymbol{x}^{(i)}) - y^{(i)})\, x_j^{(i)}$

Regular (Batch) Gradient Descent

- In Regular Gradient Descent, we should wait to calculate all costs **over all m samples** and use the average of them to modify theta.
- Suppose that we want to train a **linear (or logistic) regression** using gradient descent algorithm on ***m = 100 Million* samples**. We need to calculate a summation over 100 Million samples in EACH iteration of gradient descent!

Repeat {

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

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

}

Stochastic Gradient Descent

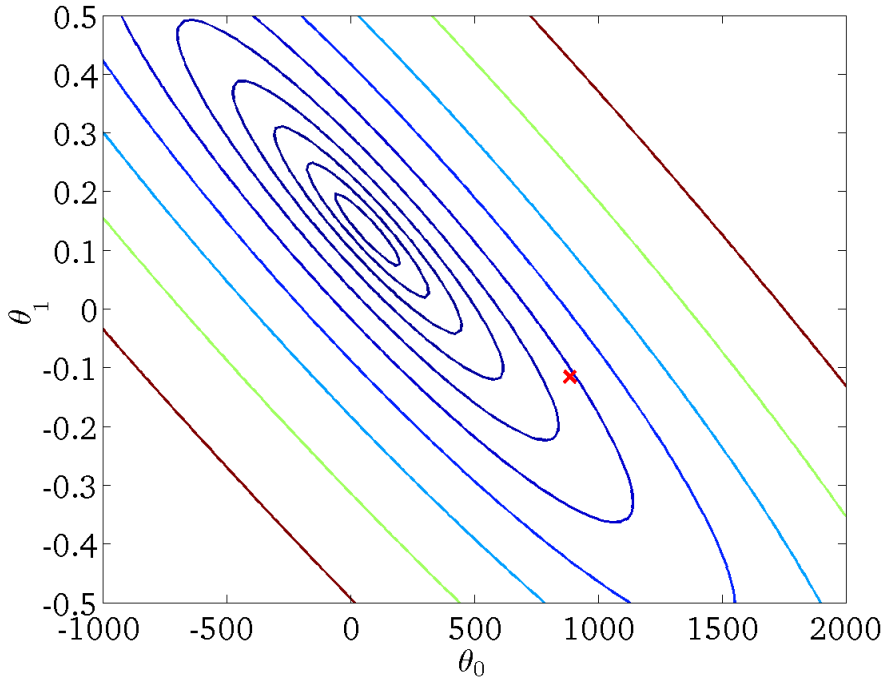
- In the Stochastic Gradient Descent, Rather than waiting to calculate all costs **over all m samples** and use the average of them to modify theta, we can start making some small progress after calculating each cost for each training sample!

Stochastic Gradient Descent

New Approach: Stochastic Gradient Descent:

1. Randomly shuffle (reorder) training examples

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



* Reference: Andrew Ng, Machine Learning, Stanford University.

Notes

- **Stochastic Gradient Descent** tries to take a small step after processing each individual training data sample. In other word, rather than waiting to go over all data samples to form the summation and make a final step, it makes a small progress using each data sample individually.

- **Stochastic Gradient Descent** uses much higher number of steps to move toward the minimum (in some cases even in wrong directions), but overall it turns out to be much faster than Batch Gradient Descent.

- In some cases, it may never reach the global minimum. But, as long as it is close enough to the global minimum, that should be good enough for most applications.

Mini-Batch Gradient Descent for Big Data

- **Regular (Batch) Gradient Descent**: Use all data samples in each iteration:

Repeat {

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

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

}

- **Stochastic Gradient Descent**: Use 1 example in each iteration:

Repeat {

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**: Use b examples in each iteration!

Mini-Batch Gradient Descent

- **Mini-batch Gradient Descent**: Use b examples in each iteration.
- b is called ***Batch Size***.

Example: $b = 10, m = 1000.$

Repeat {

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

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

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

}

}

Mini Batch Gradient Descent

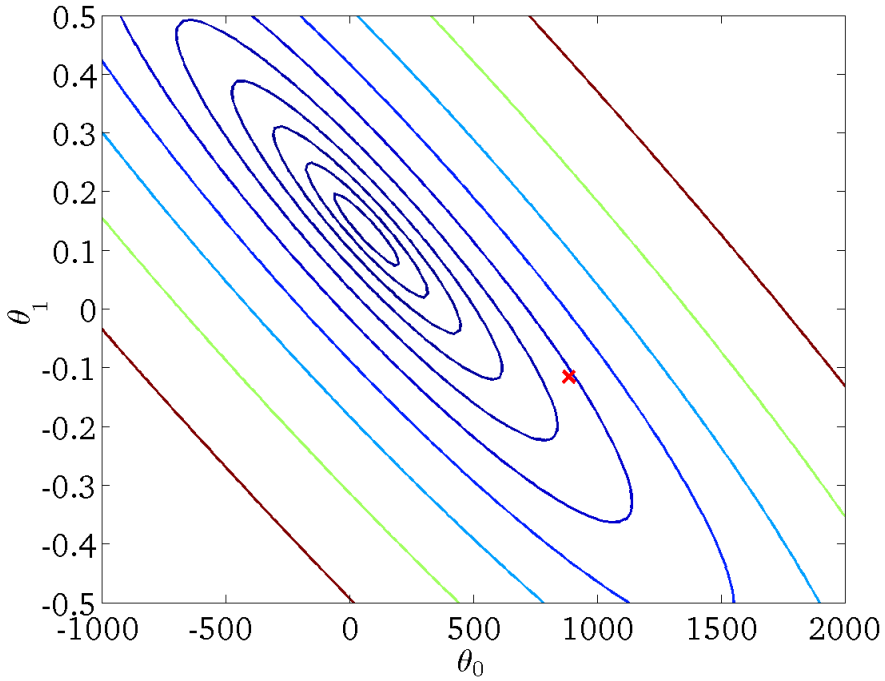
Mini-Batch Gradient Descent:

1. Randomly shuffle (reorder) training examples

2. Say $b = 10, m = 1000$.
   Repeat {
   for $i = 1, 11, 21, 31, \ldots, 991$ {

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

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



* Reference: Andrew Ng, Machine Learning, Stanford University.

Thank You!

Questions?