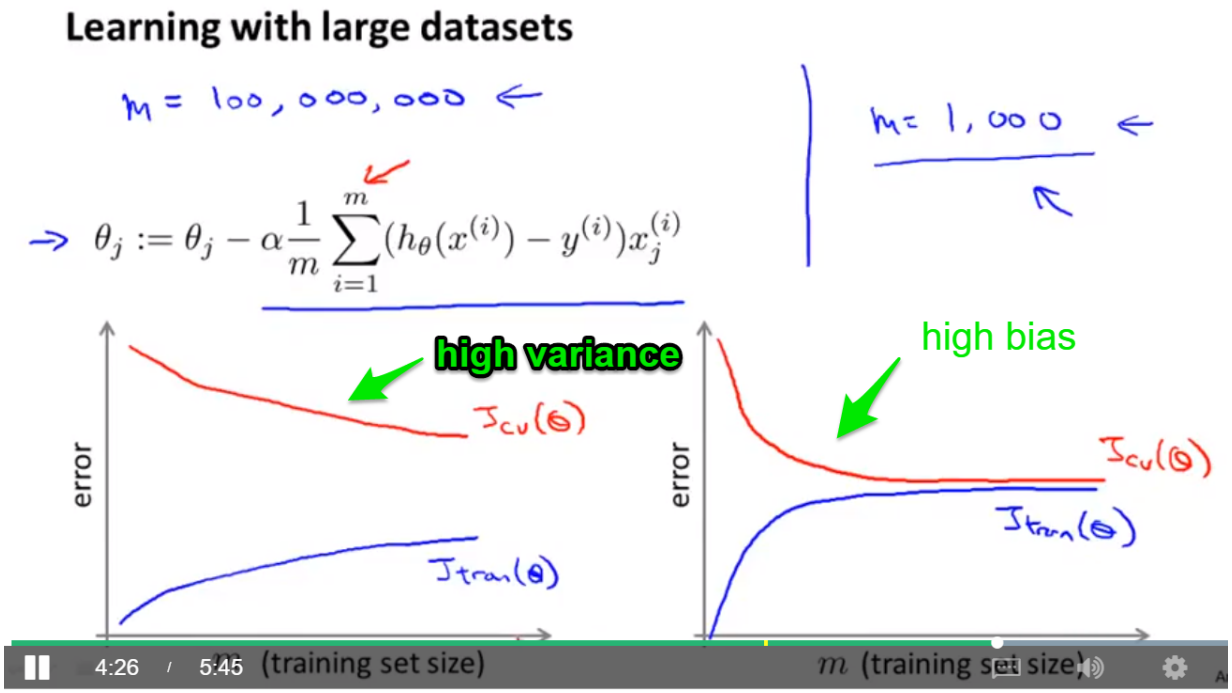
machine learning_week10

Created: 18/12/02 14:58 **Updated:** 18/12/08 16:35

Gradient Descent with Large Data set

- before implementing large data set, try small data set first!
  - verify small data set whether it work or not.
    - by plotting learning curve. when **variance** is **high**, we should try large data set.



- facing problem: **computational expensive for gradient**

Stochastic Gradient Descent

- what the GD we used in the past could be called **batch gradient descent.**
- main idea: **using subset to lower computational cost.**
- **algorithm**
  1. randomly shuffle data set
  2. loop for a subset of examples and iterate to descend gradient by:

$$\text{2. For } i = 1 \ldots m$$

$$\Theta_j := \Theta_j - \alpha(h_\Theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

- attribute:
  - may not converge into global minimum but would be close.
  - though we still have to look through whole data set, but sometimes before going through, we **already get convergent result**. This let us can stop training earlier.

Mini-Batch Gradient Descent

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

- just using subset to compute too~

- and here comes a **hyper-parameter** , b would be hard to choose. Pro.Wang suggested that b could be 10.

  Typical values for b range from 2-100 or so.

  For example, with b=10 and m=1000:

  Repeat:

  For $i = 1, 11, 21, 31, \ldots, 991$

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

- advantages: use vectorized implementations to parallel computing.

- **tuned learning rate α / check whether GD is converging**
  - facing problem:
    - in small data set we compute J(Θ) each iteration and after training, we plot the J(Θ) to check
    - however computing J(Θ) need to **scan through whole data set**, when our data set is large. the cost is unacceptable.
  - method:
    - instead of computing J(Θ), we compute **cost(Θ,(x^i,y^i))**

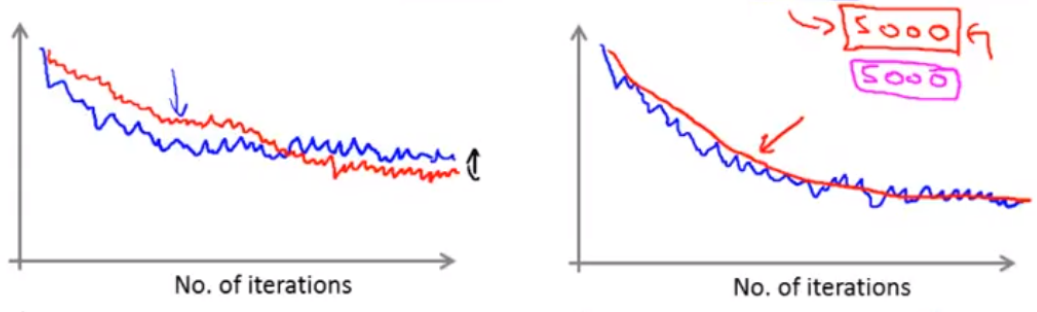
      → Stochastic gradient descent:

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

      → During learning, compute $cost(\theta, (x^{(i)}, y^{(i)}))$ before updating θ using $(x^{(i)}, y^{(i)})$.

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

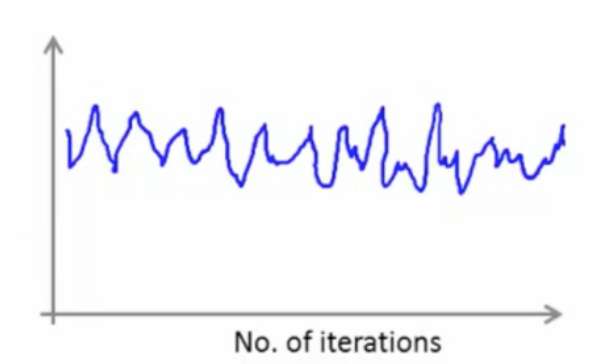
      changeable
    - then plot the **average_cost-iteration** to examine.
    - pic may like this:(due to average, there are **noise** in pic)
      - successfully implement:

      - bull shit!

      - sometimes:



-
- we need to try a large iterations like 1000→5000, to see whether the problem caused by noise or divergence.

Summary:

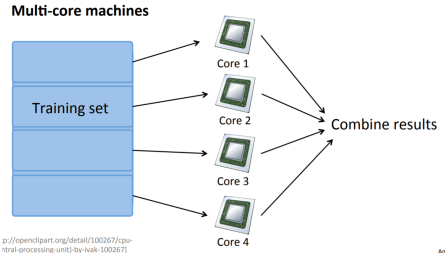
Algorithm	Large *m*	Out-of-core support	Large *n*	Hyperparams	Scaling required	Scikit-Learn
Normal Equation	Fast	No	Slow	0	No	LinearRegression
Batch GD	Slow	No	Fast	2	Yes	n/a
Stochastic GD	Fast	Yes	Fast	≥2	Yes	SGDRegressor
Mini-batch GD	Fast	Yes	Fast	≥2	Yes	n/a

Online Learning

- facing **continuous data stream.** (e.g. shopping online) It's not smart to train the model periodically. Instead, we need to update our model **instantly.**
- **main idea:**
  - **don't save data. Use and discard.**
  - **update θ by feeding example.**
- **algorithm**
  - like SGD, just compute GD with one example.
  - for instance,linear regression :
    - Θ: $\Theta-\alpha(h(X)-Y))X$
    - here y can be user behavior like whether click or not.
- advantages:
  - save place (data used as one-time)
  - dynamic (sensitive to user preference changing)
- more examples:
  1. improve CTR （Click-Through-Rate）
  2. contents recommendation (combining with collaborative filter)
  3. Commodity pricing

Map and Reduce

- split the task and distribute it
- algorithm is MapReduceable: it can be *expressed as computing sums of functions over the training set*.



-
  - on multiple cores or machines