Mini Batch Gradient Descent

Mini-batch gradient descent can sometimes be even **faster than stochastic gradient descent**. To summarize our approaches so far:

- **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

Instead of using all m examples as in batch gradient descent, and instead of using only 1 example as in stochastic gradient descent, we will use some in-between number of examples b.

- b = mini-batch size.

So just like batch, except we use tiny batches

- Typical values for b range from 2-100 (10 maybe) or so.

The idea is that rather than using one example at a time or m examples at a time we will use b examples at a time.

- e.g. Get b = 10 examples
  - $(x^{(i)}, y^{(i)}), \ldots, (x^{(i+9)}, y^{(i+9)})$
  - Perform gradient descent update using the ten examples
    - $\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$

Mini-batch Gradient Descent algorithm

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

- **Repeat: {**
  - **For i = 1, 11, 21, 31, $\ldots$, 991 {**
  - $\theta_j := \theta_j - \alpha \frac{1}{10} \sum_{k=i}^{i+9} (h_\theta(x^{(k)}) - y^{(k)}) x_j^{(k)}$
  - **(for every j = 0, $\ldots$, n)**
- **} }**

We're simply summing over ten examples at a time. The advantage of computing more than one example at a time is that we can use vectorized implementations over the b examples.

- Compared to batch gradient descent this allows us to get through data in a **much more efficient way**
  - After just b examples **we begin to improve our parameters**
  - Don't have to update parameters after every example, and **don't have to wait until we cycled through all the data**

Mini-batch gradient descent vs. stochastic gradient descent

Why should we use mini-batch? In particular, Mini-batch gradient descent is likely to outperform Stochastic gradient descent only if we have a **good vectorized implementation**.

- Can partially parallelize our computation (i.e. do 10 at once)

- By using appropriate vectorization to compute the rest of the terms, we can sometimes partially use the good numerical algebra libraries and **parallelize our gradient computations over the b examples**.
- Whereas if **we were looking at just a single example of time with Stochastic gradient descent** then just looking at one example at a time their isn't much to parallelize over.

One disadvantage of Mini-batch gradient descent is that there is now **the extra parameter b**, the Mini-batch size which **we may have to fiddle with**, and which may therefore **take time**. But if we have a good vectorized implementation this can sometimes **run even faster** that Stochastic gradient descent.

Video Question: Suppose you use mini-batch gradient descent on a training set of size m, and you use a mini-batch size of b. The algorithm becomes the same as batch gradient descent if:

- b = 1
- b = m / 2

> b = m

- None of the above

To be honest, stochastic gradient descent and mini-batch gradient descent are just **specific forms** of batch-gradient descent

- For mini-batch gradient descent, b is somewhere in between 1 and m and we can try to optimize for it.