mini-batch gradient descent

$$X = \left[x^{(1)} \quad \cdots \quad x^{(k)} \quad | \quad x^{(i+1)} \quad \cdots \quad x^{(2k)} \quad | \quad \cdots \quad | \quad x^{(i)} \quad \cdots \quad x^{(m)} \right] \tag{1}$$

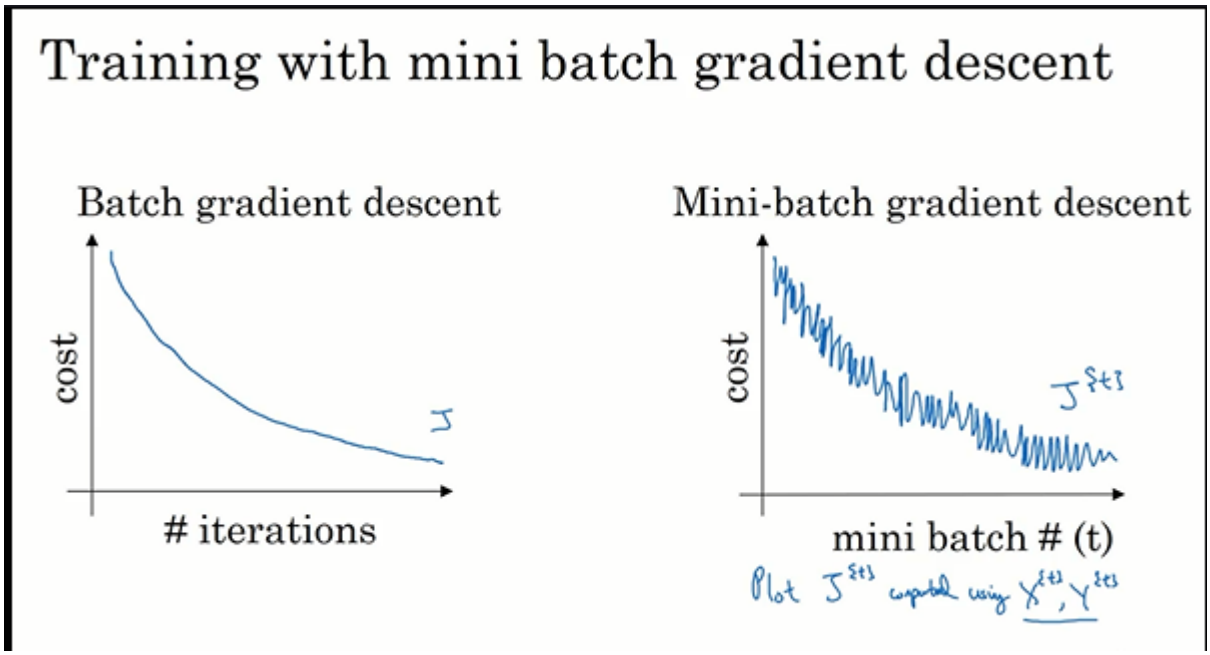
$$Y = \left[y^{(1)} \quad \cdots \quad y^{(k)} \quad | \quad y^{(i+1)} \quad \cdots \quad y^{(2k)} \quad | \quad \cdots \quad | \quad y^{(i)} \quad \cdots \quad y^{(m)} \right] \tag{2}$$

- mini-batches t : $X^{\{t\}}(n_x, mini-batch\ size)$, $Y^{\{t\}}(1, mini-batch\ size)$

  $\underbrace{x^{(1)} \cdots x^{(k)}}_{X^{\{1\}}} \quad \cdots \quad \underbrace{x^{(i)} \cdots x^{(m)}}_{X^{\{j\}}}$

  $\underbrace{y^{(1)} \cdots y^{(k)}}_{Y^{\{1\}}} \quad \cdots \quad \underbrace{y^{(i)} \cdots y^{(m)}}_{Y^{\{j\}}}$
- do not need to wait till the whole giant vector finish computing
- **epoch**: one pass through the deep network

Training with mini-batch gradient descent



Choosing the mini-batch size

- If minibatch size = m : Batch gradient descent
  - take low-noise and large steps
  - too long per iteration
- If minibatch size = 1 : Stochastic gradient decent
  - oscillating around the optimal point
  - lose speedup from vectorization
- In practice: between 1 and m

- Small training set: use batch gradient descent
- typical mini-batch size: **64, 128, 256, 512, ...**
    - try several values to see if it optimizes the model
- make sure minibatch size fit in CPU\GPU memory