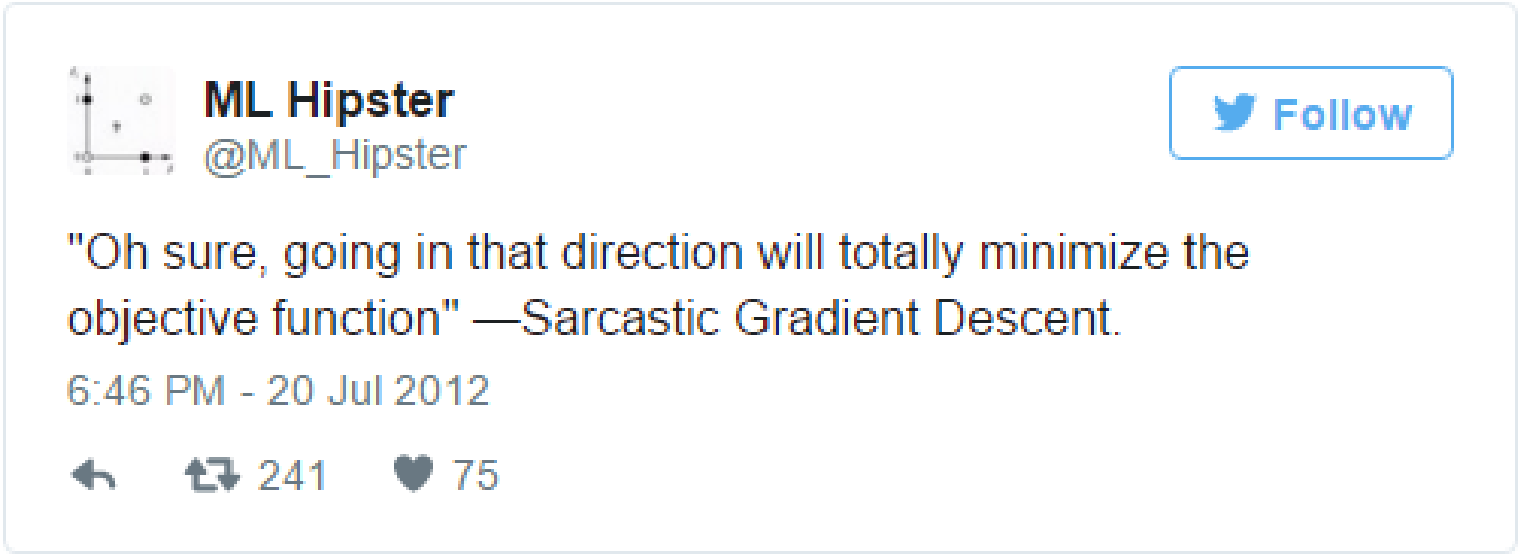
Intro to Optimizing Neural Networks



Material from Andrej Karpathy
http://cs231n.github.io/neural-networks-2/#init

Mini-Batch Stochastic Gradient Descent

- Instead of minimizing the cost function $\sum_{i=1}^{M} C(y^{(i)}, f_\theta(x^{(i)}))$, make a step along the gradient with respect to just a few example
  - Repeat:
    - Select random mini-batch S of training examples (size e.g. 50, but could be 1)
    - $\theta \leftarrow \theta - \alpha \frac{\partial}{\partial \theta} \sum_{i \in S} C(y^{(i)}, f_\theta(x^{(i)})$
- (Perhaps) helps avoid bad local minima because the direction of the current gradient changes all the time
  - (Note: in deep neural networks, we're not so worried about bad minima)
- Don't need to store all the data in RAM
  - Useful a lot of the time!

Weight Initialization

- Extremely important for Multilayer Neural Networks!
- *Not* all zeros
  - If all the neurons in a layer are the same, they can only change in the same direction by the same amount
- Small random numbers
  - Not *too* small, since that might case the gradient to be small
  - Called “symmetry breaking”
  - Good enough for CSC321
- Heuristic: random numbers that depend on the number of incoming weights:
  - $N(0,1)/\sqrt{n}$. This makes the inputs to all the units initially be on approximately the same scale
- Can set biases to 0
  - Symmetry breaking provided by the weight initialization

ML Hipster
@ML_Hipster

Follow

Everyone is all big data this and online that. My methods are small batch: they only handle a few instances but really look at them, y'know?

6:28 PM - 16 Aug 2012

17 6