Getting Started

Wednesday, January 27, 2016 1:38 PM

- MNIST dataset: 60k examples for training set(of which, 10k are for validation), 10k for testing.
  - Every example is a fixed size image of 28x28px
    - Each pixel is represented by a value between 0 and 1(normalized from the 0 to 255 vals)
  - Tuple of 3 lists(training, validation, testing)
    - Each list is a pair of images and a pair of class labels(numbers the images stand represent)
      - Each image represented as a 1d numpy array of 784 float vals
- Theano tips:
  - Minibatches used with an index
    - Minibatches are loaded into the GPU memory all at once into a shared variable because copying things to the GPU memory adds a large operational overhead - then they are selected with an index
    - Use different shared variables for the labels and the data - because they are different types of data.
    - Use different shared variables for the 3 different sets
  - When storing data in GPU, you need to use floats
    - dtype should be theano.config.floatX
  - If not enough memory on the GPU, you can chunk your data further
- Notation:
  - D: Dataset
  - D_{train}, D_{valid}, D_{test} sets
  - Each dataset is an indexed set of pairs ($x^{(i)}, y^{(i)}$)
  - Superscripts used to distinguish training sets:
    - $x^{(i)} \in \mathcal{R}^D$
      - Is the ith training example of dimensionality D
    - $y^{(i)} \in \{0, ..., L\}$
      - Is the ith label assigned to input $x^{(i)}$
      - $y^{(i)}$ can have other types

    Math Conventions

    - W: upper-case symbols refer to a matrix unless specified otherwise
    - W_{ij}: element at i-th row and j-th column of matrix W
    - $W_{i\cdot}, W_i$: vector, i-th row of matrix W
    - $W_{\cdot j}$: vector, j-th column of matrix W
    - b: lower-case symbols refer to a vector unless specified otherwise
    - b_i: i-th element of vector b
  - **List of Symbols and acronyms**
    - D: number of input dimensions.
    - $D_h^{(i)}$: number of hidden units in the i-th layer.
    - $f_\theta(x)$, $f(x)$: classification function associated with a model $P(Y|x,\theta)$, defined as $\operatorname{argmax}_k P(Y=k|x,\theta)$. Note that we will often drop the θ subscript.
    - L: number of labels.
    - $\mathcal{L}(\theta, \mathcal{D})$: log-likelihood $\mathcal{D}$ of the model defined by parameters θ.
    - $\ell(\theta, \mathcal{D})$ empirical loss of the prediction function f parameterized by θ on data set $\mathcal{D}$.
    - NLL: negative log-likelihood
    - θ: set of all parameters for a given model

Gradient-Based Learning

Wednesday, January 27, 2016 8:26 PM

- $C(\theta) = \frac{1}{n}\sum_{i=1}^{n} L(f_\theta, z_i)$
  - Cost function is the average/expectation of a loss function(training loss)
  - θ : parameter vector
  - $C(\theta)$ is a scalar value which we want to minimize
  - $z = (x, y)$
  - $f_\theta(x)$ is a prediction of y, indexed by the parameters θ
  - The gradient of $C(\theta)$ when θ is a single scalar is:
    - $\frac{\partial C(\theta)}{\partial \theta}$
    - When θ is a vector, we hold other parameters fixed and find the change and result
- Gradient descent:
  - Ideally, we want to find the values at which:
    - $\frac{\partial C(\theta)}{\partial \theta} = 0$
  - Because we usually can't find the minima, we aim to find the local minima through local descent; iteratively modifying θ as to decrease $C(\theta)$, until we cannot anymore
  - $\theta^{k+1} = \theta^k - \epsilon_k \frac{\partial C(\theta^k)}{\partial \theta^k}$
    - Ordinary gradient descent
    - ϵ_k is the learning rate
    - θ^k represents the parameters at the kth iteration
- Stochastic gradient descent:
  - $\theta^{k+1} = \theta^k - \epsilon_k \frac{\partial L(\theta^k, z)}{\partial \theta^k}$
    - z is next example from training set
  - Works because C is an average of the losses
  - Much faster because we make constant changes to the parameters after each example
- Mini-batch gradient descent:
  - Average a small batch of the training set in order to get the direction
  - Between batch gradient descent and stochastic gradient descent in functionality