Deep Learning Workshop - TensorFlow

July 12, 2016

Exercise 1 - Simple binary-classification on 2d-data

For k classes, we can define multinomial logistic regression as

$$P[c = j] = \frac{e^{\beta_j^\top \mathbf{x}}}{\sum_i e^{\beta_i^\top \mathbf{x}}}$$

Instead of k individual terms $\beta_j^\top \mathbf{x}$, we can think of a vector $\mathbf{W}\mathbf{x} + \mathbf{b} \in \mathbb{R}^k$ that parametrizes the underlying normalization function. In the language of neural networks, we use a linear layer on the input $\mathbf{x}$.

In tensorflow, we can express this using a softmax. The classification loss can be measured in terms of cross-entropy loss.

$$\text{input} \rightarrow \text{linear} \rightarrow \text{softmax} \rightarrow \text{cross-entropy loss}$$

Your task

- Implement the suggested graph and run the training routine. You should see the train and the test error decrease.
- Start tensorboard using

  `tensorboard --logdir=/your/logdir/path`

  [1] and go to `http://localhost:6006` (Chrome recommended but Firefox should work, too). Open the Graph section and try to follow the flow of information in the main graph. Match it to your code.
  The "gradients" box under auxiliary nodes can be expanded to show the graph necessary to compute the gradients.
- Add another layer on top of the first and retrain. Does the performance increase?
- To optimize the parameters more sensibly, introduce a linear learning rate decay of the form

  $$\text{lr} = \text{initial\_lr}\left(1 - \frac{t}{t_{\text{max}}}\right)$$

  where t is the current time (e.g. global_step provided by `optimizer.minimize(loss, global_step = global_step)` and t_{max} is the total number of steps you are planning to do in all epochs. initial_lr is a tuning parameter.
- If time allows, write a mini-batch version of your program. Hint: Typically the first dimension is used as the "batch dimension".

[1] You can set the `logdirPath` variable at the beginning of the file