Assignment 1
IFT 6268 Winter 2016

Date posted: February 16, 2016
Due: February 24, 2016, at the *beginning* of class

---

1. (1/15) Using the definition

$$\text{logsumexp}(a_1, \ldots, a_K) = \log\left(\sum_i \exp(a_i + A)\right) - A$$

   show that $\text{logsumexp}(a_1, \ldots, a_K) = \log \sum_i \exp(a_i)$

2. (1/15) Derive a numerically stable expression for the logistic sigmoid $\sigma(a) = \frac{1}{1+\exp(-a)}$

3. (2/15) Assume $s(t)$ to be a *real valued* signal with spectrum $S(l)$. Prove or disprove:

$$S(l) = \bar{S}(T - l)$$

4. (6/15) In this and in the following question you will train and apply a simple object detector using the CIFAR-10 dataset by training a convolutional network whose fully connected layers are implemented as 1×1-convolutions.

   We will use a variation of the VGG-network proposed in "Very Deep Convolutional Networks For Large-scale Image Recognition" (Simonyan, Zisserman; 2015).

   What you need to do:

   - Download the CIFAR-10 dataset from

     `http://www.cs.toronto.edu/~kriz/cifar.html`

     Do not normalize the data in any way, except for using single precision floating point numbers rather than integers to represent the images.
   - Using a back-prop package of your choice, train a convolutional network whose convolution filters are all of size 3×3. Use a network with the following structure:
     (a) conv-layer with 3 inputs (RGB), 64 outputs (filter size is thus $64 \times 3 \times 3 \times 3$)
     (b) 2×2 max-pooling layer
     (c) conv-layer with 64 outputs, 128 outputs (filter size $128 \times 64 \times 3 \times 3$)
     (d) 2×2 max-pooling layer
     (e) conv-layer with 128 inputs, 256 outputs
     (f) conv-layer with 256 inputs, 256 outputs
     (g) 2×2 max-pooling layer

(h) fully connected layer with 256 inputs, 1024 outputs
(i) fully connected layer with 1024 inputs, 1024 outputs
(j) softmax layer for classification: 1024 inputs, 10 outputs

- If you use theano, you can use the ops

  `T.nnet.conv.conv2d`

  to define the convolution layers and

  `theano.tensor.signal.downsample.max_pool_2d`

  to define the pooling layers.
- All convolution layers should be dense (no strides), and all pooling layers should be non-overlapping.
- All layers, except for the pooling layers and for the last (softmax-)layer should use ReLU-nonlinearities.
- Define the fully connected layers as 1×1-convolutions. This is necessary to apply the model as a detector in the next question. You will first have to determine the size of the last layer before the first fully connected layer when CIFAR images (32×32) are the model inputs.
- Augment the training dataset by adding for each training case a horizontally flipped version.
- Train the network using gradient descent or gradient descent with momentum on 95000 randomly chosen examples from the resulting training dataset. Use the remaining 5000 examples for validation.
- Use *batch-normalization* (http://arxiv.org/abs/1502.03167) on the last layer activations (immediately before computing the softmax) when training the network. Since you will implement this layer as a convolution, you need to use the convolutional version of batch-normalization (as described in that paper), where you compute statistics across training cases *and* spatial positions. At test-time, replace batch-means and -standard deviations using statistics that you compute from the training set.
- It will be necessary to experiment with learning rate and parameter initializations to find settings that are stable and yield good solutions.
- Train your network for 20 epochs or more.

What to hand in:

- An example image from the training set as well as its horizontally flipped version.
- A plot of the learning curve showing iterations on the x-axis and negative log-likelihood over labels on the y-axis. Make one plot showing both the training loss and the validation loss.
- The performance on the test data of the model that performs best on the validation data.

- Exactly how many parameters (floating point numbers) does your network have? How many of these are in the fully connected layers and how many are in the convolutional layers?
- Exactly how many "neurons" does your network have? How many of these are in the fully connected layers and how many are in the convolutional layers?
- Do *not* hand in any program code.

5. (5/15) Apply your network to an object detection task as follows.

   - Compute the output of your network on the following image:

     `www.iro.umontreal.ca/~memisevr/teaching/ift6268_2016/boat.jpg`

     Since your network is fully convolutional, the result should consist of 10 "feature maps", each showing probablities for one class. (But note that due to pooling, the output feature maps will be smaller than the original image.)
   - Low-pass filter the 10 probability maps by convolving them with an appropriate filter.
   - For each of the low-pass filtered probability maps, find the largest value. This will give you 10 values in total, one for each class.
   - Measure the time it takes (average over 3 runs or more) to compute the probability images for a single input image on the GPU.

   What to hand in:

   - The low-pass filtered probability images as gray-value images (10 images in total).
   - Describe and/or show the filter you used for low-pass filtering.
   - The 10 max values.
   - The timing result.