Exercise for MA-INF 2213 Computer Vision SS16
13.06.2016
Submission on 20.06.2016
Neural Networks

1. **Convolutional Neural Networks**

   Clone[1] and install the library `Lasagne` for convolutional neural networks. The library provides a python interface. You may also need to install some dependencies such as `numpy`, `scipy`, and `theano`. In this task, you are requested to train a small CNN on the USPS dataset for digit recognition. The dataset is very small, so even on a CPU the CNN can be trained quickly.

   (a) Write a function to load the data directly from the gzipped files. Use pythons `gzip` library for this task. The format of the input files is specified in the README.
   (2 Points)

   (b) Implement the `build_cnn` function. The network architecture should be as specified in the function description.
   (2 Points)

   (c) Implement the training and testing procedure as requested in the comments of the `main` function. Make sure to use the correct parameters (learning rate, momentum, ...).
   (2 Points)

   (d) Implement the `normalize` function such that the data is normalized to have zero mean and unit variance. Train the network twice, once with normalization, once without. Report the accuracy on the test set for both training runs.
   (2 Points)

   Implement your solutions in `task.py`. `TODO`s indicate where code has to be added.

   Hint: Have a look at the `mnist.py` example on github in order to solve this task.

2. **Cross entropy criterion and softmax output**

   In this task, we consider a multi-layer perceptron rather than a CNN.

   Neural networks are often trained according to the *cross-entropy* criterion. Consider a neural network with L layers $l = 1, \dots, L$. Let $\mathbf{y}^{(l)}$ denote the output of layer l and, for terms of simplicity, $\mathbf{y}^{(0)}$ the input to the network. Each layer has D_l units. Then, we have for the linear transformation in layer l

   $$\mathbf{a}^{(l)} = {\mathbf{W}^{(l)}}^T \mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}, \tag{1}$$

   where $\mathbf{W}^{(l)} \in \mathbb{R}^{D_{l-1} \times D_l}$ are the weights and $\mathbf{b}^{(l)} \in \mathbb{R}^{D_l}$ is the bias of layer l. The output of layer l is then given by

   $$\mathbf{y}^{(l)} = \sigma^{(l)}(\mathbf{a}^{(l)}) \tag{2}$$

[1] git clone https://github.com/Lasagne/Lasagne

with $\sigma^{(l)} : \mathbb{R}^{D_l} \mapsto \mathbb{R}^{D_l}$ being the non-linear function in layer l.
In the hidden layers, let $\sigma^{(l)}$ be the element-wise sigmoid function

$$\sigma_i^{(l)}(\mathbf{a}^{(l)}) = \frac{1}{1 + \exp\left(-a_i^{(l)}\right)}, \quad l \in \{1, \ldots, L-1\}, \tag{3}$$

but in the output layer, the softmax function is used:

$$\sigma_i^{(L)}(\mathbf{a}^{(L)}) = \frac{\exp\left(a_i^{(L)}\right)}{\sum_{d=1}^{D_L} \exp\left(a_d^{(L)}\right)}. \tag{4}$$

The natural training criterion for these kind of neural networks is the *cross-entropy* criterion, which aims at the minimization of

$$F(\Lambda) = \frac{1}{N} \sum_{n=1}^{N} F_n(\Lambda), \quad F_n(\Lambda) = -\log\left(p(c_n|\mathbf{x}_n)\right), \tag{5}$$

where $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)$ denotes the training data. The dimension of the output layer D_L equals the number of classes for the classification task and if $\mathbf{x}$ is the input to the network, $y_c^{(L)} = p(c|\mathbf{x})$.

(a) Assume a neural network with the above structure, i.e. a network with L layers, the sigmoid function in the hidden layers and softmax in the output layer. The training criterion is cross-entropy. Calculate the *error signals* of the output layer L,

$$\frac{\partial F_n}{\partial a_i^{(L)}} \quad (1 \leq i \leq D_L). \tag{6}$$

(2 Points)

(b) Now, calculate the error signals of each hidden layer,

$$\frac{\partial F_n}{\partial a_i^{(l)}} \quad (1 \leq l \leq L-1, \quad 1 \leq i \leq D_l). \tag{7}$$

(2 Points)

(c) Finally, give formulas for the derivatives with respect to the weights and biases,

$$\frac{\partial F}{\partial w_{ij}^{(l)}} \quad \text{and} \quad \frac{\partial F}{\partial b_j^{(l)}} \quad (1 \leq l \leq L) \tag{8}$$

using the error signals.
(2 Points)

3. **Meaning of the cross-entropy criterion**

The squared error criterion aims to minimize (as the name says) the squared error of the network output compared to the ideal output defined by the training data. Which quantity is minimized/maximized by the cross-entropy criterion?

(2 Points)

4. **Equivalence of Gaussian Models and Neural Networks**

   With neural networks, usually class posteriors $p(c|\mathbf{x})$ are modeled. Gaussian models, in contrast, can be used to model probabilities $p(\mathbf{x}|c) = \mathcal{N}(\mathbf{x}|\mu_c, \Sigma_c)$. Using Bayes' Theorem, it is also possible to model class posteriors with Gaussian models:

$$p(c|\mathbf{x}) = \frac{p(c)\mathcal{N}(\mathbf{x}|\mu_c, \Sigma_c)}{\sum_{\tilde{c}} p(\tilde{c})\mathcal{N}(\mathbf{x}|\mu_{\tilde{c}}, \Sigma_{\tilde{c}})}. \tag{9}$$

   Assume a neural network without hidden layer and a softmax output layer with C (number of classes) many output nodes. As input to the network, consider a quadratic expansion of the original input vector $\mathbf{x}$, i.e. $\hat{\mathbf{x}}$ contains all elements of $\mathbf{x} \in \mathbb{R}^D$ and all elements of $\mathbf{x}\mathbf{x}^T$. So, $\hat{\mathbf{x}} \in \mathbb{R}^{D+D\cdot D}$, and the network realizes the function

$$p(c|\hat{\mathbf{x}}) = \text{softmax}(\mathbf{W}^T\hat{\mathbf{x}} + \mathbf{b}). \tag{10}$$

   Show that the model (9) can be transformed into a model of the form of (10).

   Remark: The transformation is also possible in the other direction, i.e. from (10) to (9). Hence, both models are in fact equivalent.

   (4 Points)

If you write neatly you may upload a scan of your solutions for the theoretical exercises. Of course, you may as well use LaTeX if you like.