Engineering Tripos Part IIB FOURTH YEAR

Module 4F12: Computer Vision and Robotics

Examples Paper 4: Solutions

Straightforward questions are marked †
Tripos standard (but not necessarily Tripos length) questions are marked ∗

1. ∗ *Neural networks*

"Show that the derivatives of the cost function are :$\frac{\mathrm{d}}{\mathrm{d}\mathbf{w}}G(\mathbf{w}) = -\sum_{\mathbf{n}}(\mathrm{t}^{(\mathrm{n})} - \mathrm{x}^{(\mathrm{n})})\mathbf{z}^{(\mathbf{n})}$"

First note that application of the chain rule yields,

$$\frac{\mathrm{d}}{\mathrm{dw_k}}G(\mathbf{w}) = -\sum_{\mathbf{n}} \frac{\mathrm{t}^{(\mathrm{n})} - \mathrm{x}^{(\mathrm{n})}}{\mathrm{x}^{(\mathrm{n})}(1 - \mathrm{x}^{(\mathrm{n})})}\frac{\mathrm{dx}^{(\mathrm{n})}}{\mathrm{dw_k}}.$$

Then we find,

$$\frac{\mathrm{dx}^{(\mathrm{n})}}{\mathrm{dw_k}} = x^{(n)}(1 - x^{(n)})z_k^{(n)}.$$

Combining the above recovers the desired result.

"Interpretting the output of the network as $x_i = p(\mathrm{t_i} = 1|\mathbf{w}, \mathbf{z}) = \frac{\exp(\mathbf{w_i}^{\mathrm{T}}\mathbf{z})}{\sum_{\mathbf{j}}\exp(\mathbf{w_j}^{\mathrm{T}}\mathbf{z})}$ write down a cost-function for training this network based on the log-probability of the training data given the weights $\mathbf{w}$ and inputs $\{\mathbf{z^{(n)}}\}_{\mathbf{n=1}}^{\mathbf{N}}$."

A sensible cost function is given by the log-probability of the class labels given the inputs:

$$G(\mathbf{w}) = -\sum_{\mathbf{n}}\sum_{\mathbf{i=1}}^{\mathbf{I}} \mathrm{t_i^{(n)}} \log \mathrm{x_i}(\mathbf{z^{(n)}}; \mathbf{w}).$$

"What is the relationship between this network and the one described in the first part of this question?"

let the number of classes $I = 2$ we note that

$$x_1 = \frac{\exp(\mathbf{w_1^{\mathrm{T}}z})}{\exp(\mathbf{w_1^{\mathrm{T}}z}) + \exp(\mathbf{w_2^{\mathrm{T}}z})} = \frac{1}{1 + \exp(-(\mathbf{w_1} - \mathbf{w_2})^{\mathrm{T}}\mathbf{z})} = 1 - x_2$$

which is identical to the above when $\mathbf{w} = \mathbf{w_1} - \mathbf{w_2}$

now check that the cost function we proposed is identical too

$$G(\mathbf{w}) = -\textstyle\sum_{\mathbf{n}} \left[\mathrm{t}_1^{(\mathrm{n})} \log \mathrm{x}_1(\mathbf{z}^{(\mathbf{n})};\mathbf{w}) + \mathrm{t}_2^{(\mathrm{n})} \log \mathrm{x}_2(\mathbf{z}^{(\mathbf{n})};\mathbf{w})\right]$$
$$= -\textstyle\sum_{n} \left[\mathrm{t}_1^{(\mathrm{n})} \log \mathrm{x}_1(\mathbf{z}^{(\mathbf{n})};\mathbf{w}) + (\mathbf{1} - \mathrm{t}_1^{(\mathrm{n})}) \log(1 - \mathrm{x}_1(\mathbf{z}^{(\mathbf{n})};\mathbf{w}))\right]$$

2. $*$ *Convolutional neural networks*

"Show that the derivatives of the objective function with respect to the convolutional weights, w_k, can themselves be computed efficiently using convolutions."

First apply the chain and product rules:

$$\frac{\mathrm{d}}{\mathrm{dw_k}}G(\mathbf{w}) = \sum_{\mathbf{n}} \frac{\mathrm{dG}(\mathbf{w})}{\mathrm{dx}^{(\mathrm{n})}} \sum_{\mathbf{j}} \frac{\mathrm{dx}^{(\mathrm{n})}}{\mathrm{dy_j^{(n)}}} \frac{\mathrm{dy_j^{(n)}}}{\mathrm{da_j^{(n)}}} \frac{\mathrm{da_j^{(n)}}}{\mathrm{dw_k^{(n)}}}$$

then compute the component parts:

$$\frac{\mathrm{dG}(\mathbf{w})}{\mathrm{dx}^{(\mathrm{n})}} = -\frac{\mathrm{t}^{(\mathrm{n})} - \mathrm{x}^{(\mathrm{n})}}{\mathrm{x}^{(\mathrm{n})}(1-\mathrm{x}^{(\mathrm{n})})}, \quad \frac{\mathrm{dx}^{(\mathrm{n})}}{\mathrm{dy_j^{(n)}}} = x^{(n)}(1-x^{(n)})v_j.$$

$$\frac{\mathrm{dy_j^{(n)}}}{\mathrm{da_j^{(n)}}} = \frac{\mathrm{df(a_j^{(n)})}}{\mathrm{da_j^{(n)}}} = f'(a_j^{(n)}), \quad \frac{\mathrm{da_j^{(n)}}}{\mathrm{dw_k}} = z_{j-k}^{(n)}$$

finally combine everything together

$$\frac{\mathrm{dG}(\mathbf{w})}{\mathrm{dx}^{(\mathrm{n})}} = -\sum_{n}(\mathrm{t}^{(\mathrm{n})} - \mathrm{x}^{(\mathrm{n})}) \sum_{\mathrm{j}} \mathrm{v_j f'(a_j^{(n)}) z_{j-k}^{(n)}}$$

which involves convolutions between $v_j f'(a_j^{(n)})$ and a reversed version of $z_j^{(n)}$

3 a) "Provide a probabilistic interpretation for the network's output and use this to justify the form of the objective function."

Answer

Let the probability that a person's age is t^n given the image of their face $Z^{(n)}$ and the network weights V, W be given by a Gaussian $p(t^{(n)}|Z^{(n)}, V, W) = G(t^{(n)}; x^{(n)}, \sigma^2)$. In this way, the output of the network $x^{(n)}(Z^{(n)}, V, W)$ is the mean of the Gaussian and σ^2 is its variance. The probability of the dataset labels given the parameters is therefore:

$$p(D|V,W) = \prod_{n=1}^{N} p(t^{(n)}|Z^{(n)}, V, W)$$
$$= \frac{1}{Z(\sigma^2)} \exp\left(-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(t^{(n)} - x^{(n)})^2\right) \tag{1}$$

Placing independent zero-mean Gaussian priors over the weights $V_{i,j}$ and $W_{i,j}$ with variance $1/\alpha$ and $1/\beta$ respectively, yields:

$$p(V, W|\alpha) = \prod_{i,j} p(w_{i,j})p(v_{i,j}) = \prod_{i,j} \frac{1}{\mathcal{Z}(\alpha, \beta)} \exp\left(-\frac{1}{2}\left(\alpha W_{i,j}^2 + \beta V_{i,j}^2\right)\right) \tag{2}$$

In this way we can interpret the objective function as relating to the probability of the weight vector given the training data, $p(W, V|D, \alpha) = \frac{1}{Z(\alpha,\beta)Z(\sigma^2)} \exp(-G(V, W))$.

b) "Describe how to train the network's convolutional weights W using gradient descent. Compute the derivative required to implement gradient descent. Simplify your expression and interpret the terms."

Answer

The gradient descent algorithm operates as follows:

i. initialise the weights (e.g. using Gaussian noise with a small variance)
ii. compute the derivative of the objective function with respect to the convolutional weights $\frac{\mathrm{dG(V,W)}}{\mathrm{dW_{i,j}}}$
iii. step down the gradient $W_{i,j} \leftarrow W_{i,j} - \eta\frac{\mathrm{dG(V,W)}}{\mathrm{dW_{i,j}}}$ (η is a user defined learning rate)
iv. loop to step (2) until $\Delta G(V, W) <$ tol

To compute the derivative we use backpropagation (aka the chain rule)

$$\frac{\mathrm{d}}{\mathrm{dW_{a,b}}} G(V, W) = \sum_{n=1}^{N}\sum_{i,j} \frac{\mathrm{dG(V,W)}}{\mathrm{dx^{(n)}}}\frac{\mathrm{dx^{(n)}}}{\mathrm{dy_{i,j}^{(n)}}}\frac{\mathrm{dy_{i,j}^{(n)}}}{\mathrm{da_{i,j}^{(n)}}}\frac{\mathrm{da_{i,j}^{(n)}}}{\mathrm{dW_{a,b}}} + \beta W_{a,b}. \tag{3}$$

where each of the terms are,

$$\frac{\mathrm{dG(V,W)}}{\mathrm{dx^{(n)}}} = \frac{1}{\sigma^2}(x^{(n)} - \mathrm{t^{(n)}}), \quad \frac{\mathrm{dx^{(n)}}}{\mathrm{dy_{i,j}^{(n)}}} = \mathrm{V_{i,j}} \tag{4}$$

$$\frac{\mathrm{dy_{i,j}^{(n)}}}{\mathrm{da_{i,j}^{(n)}}} = f'(a_{i,j}^{(n)}) \quad \frac{\mathrm{da_{i,j}^{(n)}}}{\mathrm{dW_{a,b}}} = Z_{i-a,j-b}^{(n)}. \tag{5}$$

Combining the terms together yields

$$\frac{\mathrm{d}}{\mathrm{dW_{a,b}}} G(V, W) = -\frac{1}{\sigma^2}\sum_{n=1}^{N}\left(\mathrm{t^{(n)}} - \mathrm{x^{(n)}}\right)\sum_{i,j} V_{i,j} f'(a_{i,j}^{(n)}) Z_{i-a,j-b}^{(n)} + \beta W_{ik}. \tag{6}$$

So, the derivative is simply the sum over all datapoints of the error between the predicted and true ages $\left(x^{(n)} - \mathrm{t^{(n)}}\right)$ multiplied by the sensitivity of the network's output on the convolutional weights plus a linear weight decay term. The sensitivity is a convolution between the product of the output weights and non-linearity derivative $V_{i,j}f'(a_{i,j}^{(n)})$ and the image flipped in the x and y directions $Z_{-i,-j}^{(n)}$.

c) "Describe enhancements to the architecture of the network that might improve its ability to estimate the age of a person from an image of their face."

Answer

There are lots of possible ways of improving the architecture of the network.

i. A first enhancement would use **additional sets of convolutional weights**. Currently the method only uses one set and this means that it is only able to extract a single feature (e.g. a specific oriented edge) to perform regression.

ii. A second enhancement, would use a **pooling/subsampling** stage after the non-linear stage. This would pool over a local neighbourhood and pick e.g. the max or average value. This will introduce shift invariance and reduce the number of parameters that are required in the layers above.

iii. A third enhancement would be to use a neural network with **many layers** each of which is structured as above. Together these enhancements lead to deep convolutional neural networks.

The answer should describe these enhancements in detail which is bookwork.

Richard E. Turner

4. ** CNN architectures for computer vision*

(a) "Propose a simple architecture for such a network and explain the motivation behind your choice. Write down mathematical definitions of all layers used and explain their corresponding roles. Provide a calculation of network parameters in your proposed architecture."

A simple architecture consisting of two convolutional layers (CONV) and two fully connected layers can be employed for this task:

- CONV1(K = 3×3, S = 1, C = 32, A = ReLU), OS $\rightarrow 30 \times 30 \times 32$, P = $3 * 3 * 1 * 32 + 32 = 320$. Note that input to this layer is image of shape $32 \times 32 \times 1$.
- CONV2(K = 3×3, S = 1, C = 32, A = ReLU), OS $\rightarrow 28 \times 28 \times 32$, P = $3 * 3 * 32 * 32 + 32 = 9248$.
- MAX-POOL1(K = 2×2, S = 2), OS $\rightarrow 14 \times 14 \times 32$, P = 0.
- DROPOUT1($\alpha = 0.25$), OS $\rightarrow 14 \times 14 \times 32$, P = 0.
- FC1(C = 128, A = relu), OS $\rightarrow 128$, P = $14 * 14 * 32 * 128 + 128 = 802944$.
- DROPOUT2($\alpha = 0.5$), OS $\rightarrow 128$, P = 0.
- FC2(C = 10, A = softmax), OS $\rightarrow 10$, P = $128 * 10 + 10 = 1290$.

K - kernel size, S - stride, A - activation function, C - number of convolutional filters (channels) for convolutional layer and number of output units for a fully connected layer. OS - output shape, P - number of parameters, α - probability of randomly setting a neuron to zero.

Total number of parameters[1] is: 813802. Note that bias term is included for both fully connected and convolutional layers. No padding is applied for convolutional layers. The output of DROPOUT1 layer is flattened to a 1-dimensional array before it is passed to FC1 layer.

Definitions of functions used in the aforementioned architecture are provided below. Note here x refers to an input (a multi-dimensional array) to a layer of interest and f - to an output:

- Convolution: $f_{i,j,c_{out}} = (\sum_{k,l,c_{in}} w^{c_{out}}_{k,l,c_{in}} x_{i-k,j-l,c_{in}}) + b^{c_{out}}$, where c_{in} corresponds to an input channel, $w^{c_{out}}_{d_1,d_2,c_{in}}$ corresponds to convolution kernel weights for output channel c_{out} and $b^{c_{out}}$ is a bias term for channel c_{out}.
- Fully connected layer: $f_c = \sum_i x_i w^c_i + b^c$, where w^c_i corresponds to fully connected layer weights for channel c and b^c - a corresponding bias term.
- ReLU activation function: $f_{i,j,c} = \max(0, x_{i,j,c})$. Applied to the output of a convolutional layer

[1]Note that this model achieves above 99% accuracy on MNIST dataset using 10 epochs of training and categorical cross entropy objective function.

- Softmax activation function: $f_c = \frac{e^{x_c}}{\sum_k e^{x_k}}$. Aplied to the output of a fully connected layer.
- Max-Pooling: $f_{i,j,c} = \max_{|k|<\tau,|l|<\tau} x_{i\cdot s-k,j\cdot s-l,c}$, where τ is the kernel size for pooling and s is stride.
- Dropout: $f_{i,j,c} = a_{i,j,c}x_{i,j,c}$, where $a_{i,j,c} \in \{0,1\}$ (set to 0 with probability α).

CONV1 and CONV2 layers are used to extract translation invariant features from an image. MAXPOOL1 layer performs image subsampling in order to encourage learning of feature hierarchies and to reduce number of parameters[2]. Fully connected layer FC1 forms final features of our proposed network. ReLU activation functions are used for all aforementioned layers inspired by the design of a single neuron where a linear operation is followed by a non-linear operation. This enables learning of complex (non-linear) decision boundaries. Network is finalised by adding a fully connected FC2 layer with a softmax activation function. Using this layer corresponds to using a softmax classifier on the output of FC1 layer. Finally, DROPOUT layers are used to prevent over-fitting and increase network's robustness to noisy data.

(b) "Describe how would you train such a network. Include details of: (i) data preparation, (ii) objective function, (iii) optimization algorithm, (iv) weight initialization and (iv) key steps of performance tuning."

i. Data preparation. The dataset is split into train, validation and test data portions. Network is trained on the train portion. Validation data is used for hyper-parameter tuning and for making the choice of the best architecture. Final results are reported on the test dataset.
For simplicity (also since it is not specified in the question), the following assumption is made: digit orientations and sizes in the given dataset are well aligned. In such case, rotation and zoom augmentation are not required. Also due to a large size of the dataset and simplicity of the problem other types of data augmentation could also be skipped.

ii. Objective function. Softmax classifier is used for this problem. The objective function can be defined as $L = -\sum_n \log FC2^{(n)}_{c^{(n)}_{gt}}$. Here $c^{(n)}_{gt}$ is a ground-truth label for n-th training image and $FC2^{(n)}_{c^{(n)}_{gt}}$ corresponds to the output of $c^{(n)}_{gt}$-th neuron of the FC2 layer for image n of the training dataset.

iii. Optimization algorithm. Mini-batch (stochastic) gradient descent with momentum can be used to train this network. Due to a large dataset size it would be impractical to use batch gradient descent evaluated on all training data. Momentum is used in order to reduce oscillations incurred due to noisy mini-batch gradients.

iv. Weight initialization. This network can be initialized by using a random

[2]Note that in original LeNet network a pooling layer is used in between convolutional layers as well.

initialization from a well chosen distribution (e.g. normal distribution scaled by 0.01).

v. Performance tuning. Firstly, the correctness of the implemented architecture and objective function is confirmed by attempting to overfit a small portion of the dataset (e.g. 10-100 images). This training procedure should achieve very small objective function values. Secondly, hyper-parameters such as (i) number of channels for different layers, (ii) learning rate, (iii) dropout probability α and others are explored. Finally, an ensemble of networks with best hyper-parameter setup is trained on random subsets of training data to further improve the performance.

(c) "Which state-of-the-art CNN architecture would you use if you were required to train a network to classify images (224×224 pixels each) into one of 1000 classes using ImageNet dataset (contains pairs of images and corresponding class labels)? Explain key design properties of this architecture"

VGG-16 architecture can be used for this task. It is significantly deeper and has orders of magnitude more parameters ($> 100M$) than network proposed in section (a). A more complex network is required since it operates on larger resolution images and attempts to solve a more difficult task: classification of 1000 object classes versus 10 digits. VGG-16 has simple and homogeneous architecture. It has 5 blocks of two or three 3×3 convolution layers (stride 1, padding 1, ReLU activation function). The number of channels of the first layer of each block is doubled starting from 64 (first block) and finishing with 512 channels (final block). Number of channels is kept the same within each block. At the end of each block of convolutional layers, a max-pooling layer (2×2 kernel and stride 2, no padding) is applied resulting in a down-sampling factor of 32 times (compared to input image resolution). The output of final block is flattened and passed through two fully connected layers with 4096 units each (ReLU activation). Finally, a fully connected layer with 1000 units (corresponding to number of classes) and softmax activation function is applied in order to provide class probabilities.

(d) "How would you amend the architecture of the network mentioned in section (c) and your training procedure proposed in section (b) if you were asked to build a network which performs semantic segmentation of objects of some 20 classes (e.g. dog, person, etc). You can assume that along with ImageNet dataset you also have Microsoft Common Objects in Context Dataset (containing pairs of images and corresponding per-pixel class labels) provided."

Several changes would be required:

i. Architecture. The architecture of VGG classification network is adapted by replicating key steps used in the design of FCN (Fully Connected Network) network for semantic segmentation. First, final fully connected layer is removed. Second, remaining two fully connected layers are replaced with convolutional layers (e.g. with kernel sizes of 7×7 and 1×1[3]). Third, an additional convolutional layer (kernel 1×1) with 20 channels (number of classes) is added in order to obtain a low-resolution per pixel class prediction. Finally, a backwards strided convolution (deconvolution) with softmax activation function is used to obtain per pixel class label predictions at the original input image resolution.

ii. Objective function. As with image classification task, a cross-entropy objective function can be used. However, it should be applied per pixel and not for the whole image.

iii. Weight initialization. Since ImageNet dataset is available, one can first train a VGG-16 network for the task of image classification on ImageNet dataset. Learnt parameters can then be used to initialize semantic segmentation network described above. Using this parameter initialization strategy can significantly reduce training time of the segmentation network and increase its accuracy.

iv. Data preparation. Since semantic segmentation on MSCOCO dataset is a significantly more difficult problem than digit recognition, an extensive data augmentation procedure is required. Colour normalization, rotation, horizontal flips, random crops and scales, colour jitter, randomized contrast and brightness and other data augmentation techniques can be employed. Also it is important to note that all geometric (e.g. rotation, flip, etc.) augmentation steps performed on the input image should be applied on the corresponding target per-pixel class label images.

Ignas Budvytis

[3]Note, students are not required to remember the actual kernel sizes used for this step.