Spatial statistics and image analysis. Lecture 6

Mats Rudemo

April 22, 2020

Neural nets

First:

One input layer with n_1 units and input variables $x_i, i = 1, \ldots, n_1$,

One intermediate (hidden) layer with n_2 units

One output layer with K units

For unit j in intermediate layer compute activation value a_j

$$z_j = \sum_{i=1}^{n_1} w_{ji}^{(1)} x_i + b_j^{(1)}, \tag{1}$$

$$a_j = \frac{e^{z_j}}{\sum_{j'=1}^{n_2} e^{z_{j'}}}, \tag{2}$$

for weights $w_{ji}^{(1)}$ and biases $b_j^{(1)}$. Write

$$a_j = \sigma(z_j), \quad j = 1, \ldots, n_2, \tag{3}$$

and call σ the *softmax* function.

From the hidden layer proceed to the output layer and compute output variables $f_k(k), k = 1, \ldots, K$, as

$$f_k(x) = f_k(x, \theta) = \sigma \left(\sum_{j=1}^{n_2} w_{kj}^{(2)} \sigma \left(\sum_{i=1}^{n_1} w_{ji}^{(1)} x_i + b_j^{(1)} \right) + b_k^{(2)} \right), \tag{4}$$

where $x = (x_1, \ldots, x_{n_1})$ is the vector of input variables, and θ is the parameter vector of all weights and biases.

We can add now add one more hidden layer get the output

$$f_k(x) = \sigma\left(\sum_{\ell=1}^{n_3} w_{k\ell}^{(3)} \sigma\left(\sum_{j=1}^{n_2} w_{\ell j}^{(2)} \sigma\left(\sum_{i=1}^{n_1} w_{ji}^{(1)} x_i + b_j^{(1)}\right) + b_\ell^{(2)}\right) + b_k^{(3)}\right) \tag{5}$$

Example: Consider a neural net for the MNIST database with $n_1 = 28^2 = 784$ units in the input layer, each input unit corresponding to one pixel value, and $K = 10$ corresponding to the 10 possible digits.

Output variables $f_k(x)$ sum to one.

Interpret $f_k(x, \theta)$ as the probability of digit k.

To classify images, estimate the parameter θ from a training set.

Let $\hat{\theta}$ denote the estimate of θ.

To classify an image x put

$$\hat{k}(x) = \text{argmax}_k f_k(x, \hat{\theta}) \tag{6}$$

Crucial step: estimate $\hat{\theta}$.

Parameter vector θ may contain several thousand components

Parameter estimation for neural nets, regularization

Training set $\mathcal{T}$ of $|\mathcal{T}|$ pairs (x, y)

Output $f(x, \theta)$ should approximate y

Choose loss function

Let first y and $f(x, \theta)$ be real-valued

$$L(\theta, \mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} (y - f(x, \theta))^2 \tag{7}$$

Consider now classification with K classes, example MNIST with $K = 10$

Output a probability distribution $f_k(x, \theta), k = 1, \ldots, K$,

For a pair (x, y) where k_c is the correct class, define y_k as

$$y_k = \begin{cases} 1 & \text{if } k = k_c \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

and choose the cross-entropy loss function

$$L(\theta, \mathcal{T}) = -\frac{1}{|\mathcal{T}|} \sum_{(x,y)\in\mathcal{T}} \sum_{k} y_k \log f_k(x, \theta). \tag{9}$$

We can minimize $L(\theta, \mathcal{T})$ to obtain an estimate $\hat{\theta} = \hat{\theta}(\mathcal{T})$

Problem: overfitting

To compensate introduce a regularization term $R(\theta)$, for instance

$$R(\theta) = \sum_{i=1}^{|\theta|} |\theta_i|^2, \tag{10}$$

Estimate θ by minimizing the regularized loss function

$$\mathcal{L}(\theta; \mathcal{T}, L, \lambda, R) = L(\theta, \mathcal{T}) + \lambda R(\theta), \tag{11}$$

where $\lambda \geq 0$ is a tuning parameter

To choose a tuning parameter, either use a separate validation set $\mathcal{T}'$ of pairs (x, y) or cross-validation

As alternative to the softmax activation function, one often uses a *rectified linear unit* given by

$$a = \max(0, z) \tag{12}$$

Convolutional neural nets

Let $w = (w_{k\ell})$ and $g = (g_{ij})$ be matrices, the convolution $w * g$ is defined by

$$(w * g)_{ij} = \sum_k \sum_\ell w_{k\ell}\, g_{i-k,j-\ell} \tag{13}$$

Convolutional neural nets particularly useful for images analysis

Such neural nets contain layers with convolutional transitions

$$a_{ij}^{(r+1)} = \sigma\left(\sum_{k=-p}^{p} \sum_{\ell=-p}^{p} w_{k\ell}^{(r)}\, a_{i-k,j-\ell}^{(r)}\right) \tag{14}$$

Here p usually is a small positive number

Note that we here only have $(2p+1)^2$ different weights

The same filter operation applied in different parts of $a^{(r)}$

It could consist of finding edges in an image

A convolution layer is often followed by a pooling layer reducing the layer size

Example: a maxpool operation where a layer of pixels is divided into adjacent and non-overlapping rectangles

Each rectangle is replaced in the following layer by one pixel with pixel value equal to the maximal pixel value in the rectangle

Handwritten digits. Analysis with a convolutional neural net. Table 1 gives the confusion matrix corresponding to a convolutional neural net trained on 50 000 digits and evaluated on 10 000 digits from the MNIST data set.

The neural network used consisted of six layers:

1. An input layer (28×28 pixel images)
2. A convolution layer with 20 filters of size 5×5
3. A rectified linear unit layer
4. A max pooling layer with size 2×2 pixels
5. A fully connected layer
6. A softmax layer (outputting the probability for each of the 10 classes)

The second item consists of 20 different filters working in parallel)

The resulting 20 filters are given in Figure 1

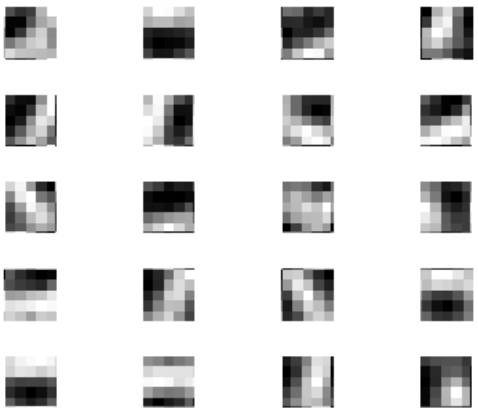

Figure 1: The 20 filters in the convolutional neural net used for identifying MNIST integers.

From the confusion matrix: the digit zero seems to be most easy to identify, estimated identification probability 99.6%

The overall estimated identification probability:
$(1130 + 1016 + \ldots 975)/10000 = 98.5\%$

True class		Estimated class											
		0	1	2	3	4	5	6	7	8	9	Sum	Percent
0	Number	1130	1	1	1	0	1	0	1	0	0	1135	11.4
	Percent	99.6	0.1	0.1	0.1	0.0	0.1	0.0	0.1	0.0	0.0	100	
1	Number	1	1016	2	1	0	1	5	4	1	1	1032	10.3
	Percent	0.1	98.4	0.2	0.1	0.0	0.1	0.5	0.4	0.1	0.1	100	
2	Number	0	3	999	0	1	0	2	4	1	0	1010	10.1
	Percent	0.0	0.3	98.9	0.0	0.1	0.0	0.2	0.4	0.1	0.0	100	
3	Number	0	1	0	974	0	1	0	0	5	1	982	9.8
	Percent	0.0	0.1	0.0	99.2	0.0	0.1	0.0	0.0	0.5	0.1	100	
4	Number	0	1	9	0	874	3	0	3	0	2	892	8.9
	Percent	0.0	0.1	1.0	0.0	98.0	0.3	0.0	0.3	0.0	0.2	100	
5	Number	3	0	0	3	1	942	0	2	0	7	958	9.6
	Percent	0.3	0.0	0.0	0.3	0.1	98.3	0.0	0.2	0.0	0.7	100	
6	Number	3	8	2	0	0	0	1012	2	1	0	1028	10.3
	Percent	0.3	0.8	0.2	0.0	0.0	0.0	98.4	0.2	0.1	0.0	100	
7	Number	0	3	2	1	2	0	5	953	2	6	974	9.7
	Percent	0.0	0.3	0.2	0.1	0.2	0.0	0.5	97.8	0.2	0.6	100	
8	Number	2	0	3	10	3	0	8	2	977	4	1009	10.1
	Percent	0.2	0.0	0.3	1.0	0.3	0.0	0.8	0.2	96.8	0.4	100	
9	Number	0	0	0	0	1	1	2	1	0	975	980	9.8
	Percent	0.0	0.0	0.0	0.0	0.1	0.1	0.2	0.1	0.0	99.5	100	
Sum	Number	1139	997	1027	984	867	963	1007	978	1011	999	10000	100
	Percent	11.4	10.0	10.3	9.8	8.7	9.6	10.1	9.8	10.1	10.0	100	

Table 1: Confusion matrix for a convolutional neural net analysis of the MNIST data set