Exam Neural Networks

Wojtek Kowalczyk
wojtek@liacs.nl
27.05.2016

It is a "closed book" exam: you are not allowed to use any notes, books, etc. For each problem you get some points (in total up to 90 points). Additionally, you get 10 points for free. The final grade for this exam is the total number of points you get divided by 10.

1) Bayes Theorem (10 points: 3+7)

a) Formulate the Bayes Theorem.

It can be formulated in several ways, eg.:
P(A|B)=P(B|A)*P(A)/P(B), or better:
P(Class|x) = p(x|Class)*P(Class)/p(x), where p( ) denotes a density function.

b) Let us suppose that a certain diagnostic classifier (perhaps a neural network) almost perfectly recognizes cancer: if you have cancer then with the probability 99% the classifier says "Cancer", if you don't have cancer, the classifier says "NoCancer", also with the probability 99%. Fortunately, cancer is not so common: it happens once per 10.000 persons. Now suppose that you have been diagnosed by the classifier as having cancer. What is the probability that you really have cancer? Write down a formula that expresses this probability.

From Bayes formula we have:
P(HasCancer|Cancer)=P(Cancer|HasCancer)*P(HasCancer)/P(Cancer).

We know that P(Cancer|HasCancer) = 0.99, P(HasCancer)=0.0001, so we need to find P(Cancer). There are two situations when we see "Cancer": either the patient has Cancer and the system correctly predicts it, or the patient has no cancer and the system makes a mistake and predicts it as Cancer. Thus:
P(Cancer)=P(Cancer, HasCancer)+P(Cancer, ¬HasCancer) =
P(Cancer|HasCancer)*P(HasCancer)+P(Cancer|¬HasCancer)*P|¬HasCancer) =
0.99*0.0001 + (1-0.99)*0.9999 = 0.0101.

Therefore, P(HasCancer|Cancer)=0.99*0.0001/0.0101=0.0098.

Roughly speaking: the positive result of the test increases our chance of having cancer by factor 100: from 1:10.000 to 1:100.

2) Modeling probability distributions (15 points: 5+5+5)

At the beginning of the course we discussed the importance of learning probability distributions from data. The following questions are related to this part of the course.

Let us consider the problem of classifying images of 3 types of objects: *A*, *B*, *C*, assuming that each image is represented by a vector of 10 real-valued features $x = (x_1,... ,x_{10})$. Additionally, let us assume that you have data on 1.000 images of each type (i.e., 3.000 images in total).

a) How could you model $p(x|A)$ with help of a histogram?
What is the main limitation of this method?

We need to split the range of each feature into a number of bins, say b, creating in this way b^10 "buckets" (or 10-dimensional bins). Then we count records (from each class) that fall into specific buckets, and then normalize these counts by dividing them by the number of records. In this way we obtain a rough approximation of P(Class|x). Unfortunately, the number of buckets is increasing exponentially fast with the dimensionality of data, so the number of records that are needed to produce reliable estimates is also increasing exponentially fast. Moreover, data binning leads to some loss of information.

b) How could you model $p(x|A)$ with help of a Gaussian distribution? Which (and how many) parameters would you have to estimate? How would you estimate them? What is the main limitation of this method?

Per class, we need to estimate the means and the covariances of 10 features. This requires calculating 10 (means) plus 55 (covariances) parameters. (The covariance matrix is 10x10, but it's symmetric). The biggest limitation of this method is the assumption that the 10-dimensional data is normally distributed - this is almost never the case!

BTW: some of you suggested to model each feature by a 1-dimensional gaussian, or even by a mixture of gaussians (with help of the EM algorithm) and then multiply them. That could work, provided features are independent of each other (given a class label). This leads to a concept of the NaiveBayes classifier. I treated such an answer as a correct one.

c) Suppose that you somehow managed to model the three density functions: $p(x|A)$, $p(x|B)$, $p(x|C)$. How would you proceed to classify an image represented by vector x? What additional information would you need?

Additionally, we need to know P(A), P(B), P(C).
Some of you suggested that we already know these 3 probabilities: for each class we have 1000 records, so all priors are equal to 1/3. I was also accepting such an answer, although samples are usually collected in a biased way and the "true priors" might be quite different.

3) Improvements of Backpropagation (10 points: 5+5)

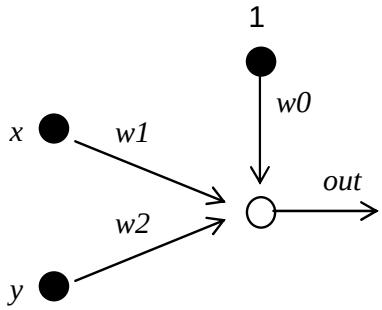
Describe in detail 2 techniques for speeding up the backpropagation algorithms:

a) Adaptive Gradient Descent,

b) Line Search.

See the slides 30, 34, 35 from NN6.pdf

4) A Simple Network (2 + 10 + 3 points)
Let us consider a single unit network with two inputs, one output, and the cubic activation function $f(a)=a^3$:



a) Express the output of the network, *out*, as a function of five variables: x, y, w_0, w_1, w_2 (write a formula).

$out(x, y, w_0, w_1, w_2) = (w_0 + x* w_1 + y* w_2)^3$

b) Derive the weight update rules for our network, assuming that we want to minimize the sum squared error, i.e., the error made by the network on input (x, y) and the target value t is given by $(out(x, y, w_0, w_1, w_2)-t)^2$. In other words, find partial derivatives of $(out(x, y, w_0, w_1, w_2)-t)^2$ over w_0, w_1, w_2 and formulate the update rules.

See slides 31-33 from NN4_5.pdf, keeping in mind that $f(net)'=3*net^2$.

c) Can this network be trained to solve the XOR-problem? Justify your answer. Here we assume that the logical values *true* and *false* are represented by +1 and -1, respectively, and the output of the network is interpreted as *true* if *out*>0 and *false* otherwise.

No, the XOR problem cannot be solved by this network. Otherwise, as $sign(a^3)$ is the same as sign(a), the XOR problem would be solvable by a single perceptron.

5) Convolutional Neural Networks (15 points: 5x2 + 3 + 2)

A) The first convolutional layer of Net5 consists of 6 feature maps, generated by 6 receptive fields of size 5x5, where each map receives inputs from a 32x32 input layer. Answer the following questions:

a) what is the size of each feature map?
28x28

b) how many connections are between the input layer and this convolutional layer (include one bias node for each receptive field)?
(5*5+1)*6*28*28

c) what is the number of trainable parameters?
6*(5*5+1)

d) how many trainable parameters would we have if the input layer was fully connected to all the nodes of the convolutional layer?
(32*32+1)*6*28*28

e) What is the role of subsampling (or pooling) layers?

Reduction of resolution (and size) of feature maps, enforcing generalization by losing some information about location of features, filtering out noise.

B) Explain the concept of "dropout". What is its main role?

See slide 36 from DeepLearning.pdf.
During each cycle of the training process a fraction of neurons (e.g., 50%) is disabled and corresponding weights are not used or affected by the training algorithm. It is a powerful technique of preventing overfitting by reduction of complex co-adaptations of neurons and forcing network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons.

C) What is data augmentation? What is the purpose of using it and how does it affect the training time and network accuracy?

See slide 34 from DeepLearning.pdf.
Data augmentation: enlarging the dataset using label-preserving transformations, such as: image translation, horizontal reflections, changing RGB intensities. The main purpose is to reduce overfitting. Network accuracy increases (on the test set); the training time increases linearly with the amount of training data.

6) Restricted Boltzmann Machines and Autoencoders (25 points: 5x5)

Let us consider an RBM with m visible nodes $x_1, ..., x_m$, n hidden nodes $h_1, ..., h_n$, and an mxn weight matrix $W=(w_{ij})$, where w_{ij} denotes the weight of the connection between x_i and h_j. For simplicity, let us ignore the biases of the visible and the hidden layer (assuming that they are 0). Additionally, let us assume that both the visible and the hidden nodes take values 0 or 1.

a) As we know, an RBM defines a joint probability distribution over all possible $2^m 2^n$ binary vectors x, h, $P(x, h)$. Define $P(x, h)$ as a function of all weights (write down a formula). Next, write down formulas for: $P(h_j|x)$ and $P(x_i|h)$.

See the formulas1 .8-1.10 and 1.16, 1.17, pages 9 and 10 of imm6284.pdf

b) What is the purpose of training the RBM on a training set X? Which function has to be optimized (give a formula)? Describe the Contrastive Divergence algorithm for training an RBM. How many multiplications are needed (by the Contrastive Divergence algorithm) to calculate a single update of weights for a single input vector x? (Assume m input nodes, n hidden nodes, no biases.)

See slides 7, 9 and 11 from NN12_RBM_Networks.pdf . The three passes up, down, and up require $3mn$ multiplications, updating weights (calculation of x^0h^0-x^1h^1) requires $2mn$ more multiplications; thus in total $5mn$ multiplications are required.

c) Describe the concept of a Deep Belief Network. Explain in detail, how such a network can be trained to recognize hand-written digits. For simplicity, let us assume that the network consists of an input layer, two RBM layers and one softmax layer.

See page 13-14 of imm6284.pdf.

d) Describe how can you build a multi-layer RBM that is trained (in an unsupervised way) on images of hand-written digits, and then used to generate even more images of digits.

Check the demo http://www.cs.toronto.edu/~hinton/digits.html that we discussed during lecture 12 on Deep Belief Networks (the DBN on page 14 of imm6284.pdf).

e) Describe the concept of a Stacked Denoising Autoencoder.
A three layer stacked denoising autoencoder (SDAE) with architecture identical to the DBN was created. The denoising autoencoder works just like the normal autoencoder except that the input is corrupted in some way, and the autoencoder is trained to reconstruct the un-corrupted input [imm6284.pdf, page 24].