DD2437 Artificial Neural Networks and Deep Architectures (annda18)

Exam 2018-03-15 at 14:00-19:00

ANSWERS

Please bear in mind that the answers below reflect the essential content. It is expected that your exam answers are a bit more elaborate at times with your brief and precise explanatory comments.

Part I (max 20p)

Question 1 (4p)

a) generalization – capability to perform regression or classification on unseen data, apply knowledge in new situations/contexts

b) artificial neural network architecture – a network graph parameterised by the number of layers, units and often by transfer function

c) learning algorithm – data driven algorithm that manipulates network's weights to fit data

d) bias-variance dilemma – problem with simultaneous minimisation of two sources of error preventing from good generalisation capabilities

e) backprop – the most common learning algorithm for multi-layer perceptrons; the essential step consists in propagating error from output to input and iteratively adjusting weights

f) dead units in competitive learning – undesirable units that never get updated in the learning process with a given dataset

g) autoassociative memory – networks storing memory patterns that can be retrieved by stimulus the same as the stored pattern

h) greedy layer-wise pretraining - an iterative process of training and building layer by layer a deep network

i) supervised vs unsupervised learning – learning with vs without labels

j) recurrent vs feed-forward networks – connectivity patterns with loops vs uni-directional layered architecture

k) Hopfield net vs Boltzmann machine – similar architecture, different learning rules, stochastic nature of Boltzmann machines

l) RBF network vs multi-layer perceptron (MLP) – differences in activation functions, architecture, weight interpretation, learning algorithm, sparseness of the hidden layer activations etc.

m) vanishing vs exploding gradients – key problem with backprop training – gradients either vanish with bacpropagating error or grow

n) shallow vs deep learning – architecture with just a couple vs a few layers

o) sequential (on-line) vs batch learning – the difference lies in how and when the weights are updated in the learning process: either following the error and gradient calculations for each sample (sequential) or collectively using the accumulated update once every epoch (batch)

p) discriminative vs generative classifier – a generative classifier models how the data was generated (joint probability over the input and output/label space) while a discriminative approach only deals with predictions that can be made based on the learned conditional probability distribution.

Question 2 (4p)

Perceptron learning always converges if the problem is linearly separable whereas the delta rule can converge without linear separability.

The main difference is the calculation of the error included in the weight update formula. For the perceptron learning rule the weighted sum (Wp) is thresholded (step function), so the error is e = target – step(Wp), whereas for the delta rule error = target – Wp. So, when the step(Wp)=target in perceptron learning rule, the error is 0 so there is no more updates. For the delta rule, as long as Wp is not equal to target, there is a weight update (the learning process continues).

The delta rule minimises mean square error so it will continue even if all training samples are correctly classified, which in consequence is more likely to provide a solution with stronger generalization power (there is no premature convergence if the training samples happen to be classified correctly; the separating hyperplane minimises mean square error and hence provides a more generalisable and intuitive solution).

You should provide a more elaborate answer than below but in short the content boils down to:

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 2 & -1 \\ -2 & 3 & 0 & 1 \end{bmatrix}$$

p1 = [1, 0, -1]^T, t1 = [1 1] ^{T},

p2 = [-1, 2, 0]^T, t2 = [0, 1] ^{T},

p3 = [1, 1, 1]^T, t3 = [0, 1] ^{T}.

After extending patterns to include bias: p1’ = [1, 0, -1, 1] ^{T} , p2’ = [-1, 2, 0, 1] ^{T}, p3’ = [1, 1, 1, 1] ^{T}

Step function is a simple threshold function with threshold = 0, so step(z)=0, z<=0 and step(z)=1, z>0

Iteration 1)

e1 = t1 -step($\mathbf{W_0}$ p1’) = [1, 1] ^{T} - step([1 + 0 – 2 – 1, -2+ 0+ 0+1] ^{T}) = [1, 1] ^{T} - step([-2, -1])= [1, 1] ^{T}

Update both **W**(1,:) and **W**(2,:) with $\Delta\mathbf{W} = \eta\, e1\, p1' = \begin{bmatrix} 0.1 & 0 & -0.1 & 0.1 \\ 0.1 & 0 & -0.1 & 0.1 \end{bmatrix}$

$$\mathbf{W}_1 = \mathbf{W}_0 + \Delta\mathbf{W} = \begin{bmatrix} 1.1 & 0 & 1.9 & -0.9 \\ -1.9 & 3 & -0.1 & 1.1 \end{bmatrix}$$

$\mathbf{W}_0$ is the initial matrix **W**

Iteration 2)

No updates, i.e. $\mathbf{W}_2 = \mathbf{W}_1$, since e2 = t2 - step($\mathbf{W}_2$ p2') = $[0, 0]^T$

Iteration 3)

e3 = t3 – step($\mathbf{W}_2$ p3') = $[0, 1]^T$ - step($[1.1+0+1.9–0.9, -1.9+3– 0.1+1.1]^T$) = $[0, 1]^T$ - step([2.1, 2.1])= $[-1, 0]^T$

Update only **W**(1,:)

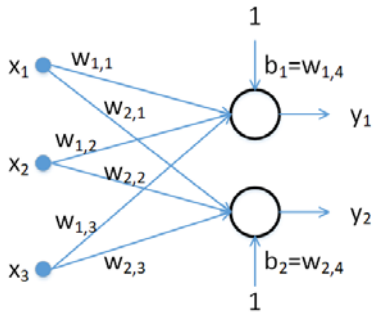
$$\Delta\mathbf{W} = \eta\, e3\, p3' = \begin{bmatrix} -0.1 & -0.1 & -0.1 & -0.1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{W}_3 = \mathbf{W}_2 + \Delta\mathbf{W} = \begin{bmatrix} 1.0 & -0.1 & 1.8 & -1.0 \\ -1.9 & 3.0 & -0.1 & 1.1 \end{bmatrix}$$

Accuracy = #correct classifications / #samples (2 out of 6 = 1/3=33%); correct classification could alternatively be defined only when both outputs match the classification target, then 1 out of 3 samples is correctly classified, which also amounts in this example to 33 %.

Not all samples are classified correctly.

The network architecture includes 3 inputs and 2 nodes with output. The bias should be included in the network diagram.



$$y_1 = \sum_{i=1}^{3} w_{1,i}\, x_i + b_1$$

$$y_2 = \sum_{i=1}^{3} w_{2,i}\, x_i + b_2$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \end{bmatrix}$$

Question 3 (2p)

Overfitting is an undesirable effect due to poor generalisation, off bias-variance balance.

Overfitting usually manifests itself by large test error (unseen data), especially when compared to training error. Techniques against overfitting

- early stopping
- regularisation, e.g. weight decay (let's count each regularisation approach as a separate technique)
- model selection with generalization error estimate obtained using cross-validation
- network growing or pruning
- dropout

Question 4 (3p)

Students should describe the concept of iterative process of updating the input weight vectors with consideration to shrinking neighbourhood determining which other nodes besides best matching units should get updated (competition with collaboration). Students should explain the concept and motivation for shrinking neighbourhood (explorative and fine-tuning/convergence phase), often accompanied by decreasing the learning rate.

topographic mapping – preserving similarity relationship when mapping from input (similarity of input vectors) to output space (proximity of the corresponding best matching units in the output grid); possible due to neighbourhood.

Kohonen maps useful for clustering or visualisation (1p)

Question 5 (4p)

Assume a Hopfield network with bipolar {1, -1} nodes and the following weight matrix, W:

$$\mathbf{W} = \begin{bmatrix} 0 & 2 & -1 & -1 & -1 \\ 2 & 0 & -1 & -1 & -1 \\ -1 & -1 & 0 & 2 & 2 \\ -1 & -1 & 2 & 0 & 2 \\ -1 & -1 & 2 & 2 & 0 \end{bmatrix}$$

x = [-1, 1, -1, 1, -1] is not a stored pattern as it maps to x = [1, -1, 1, -1, 1]. (1p)

An alternative candidate for a stored pattern is x = [-1, -1, 1, 1, 1] or x = [1, 1, -1, -1, -1]. (1p)

A weight matrix in Hopfield networks should be symmetric and should have 0s in the diagonal. (1p)

Hebbian learning is a classical approach to learning Hopfield networks. Sparseness and orthogonality promote large storage capacity in Hopfield networks. (1p)

Question 6 (3p)

Deep architectures offer more expressiveness (better use of nodes in depth than in shallow architectures to capture the complexity of modelled relationships), and more accurate performance. They also facilitate capturing hierarchical nature (which is often intrinsic) or correlations in data.

Backprop suffers from vanishing or (more rarely) exploding gradients.

Stacked autoencoders often built by greedy training of component autoencoders with backprop.

Deep belief networks based on stacked RBMs (trained with contrastive divergence)

Part II (max ~~30~~26p)

Question7

Question 8 (5p)

a) Here an MLP offers a sufficient framework for solving this regression (especially if we assume the continuous mood scale) or classification (particularly for a discrete case) problem. The number of inputs corresponds to the number of attributes and the output layer is either suited for regression (a single output) or classification (preferably 5 outputs); in both cases preferably with nonlinear activation function in the output. The number and size of hidden layers depends on the complexity of the mapping problem (unless we decide on using RBF networks). Data should be normalised across days feature by feature for example by z-scoring or scaling. Splitting the data should include a validation and a test set; for example, around 90 samples could be divided into training (60% - 54 training samples), validation (20% - 18 samples) and test (20% - 18 samples) sets.

b) Key parameters describe mainly the network architecture, the number and size of hidden layers for MLPs, or spread of the RBF activation functions in RBF networks. Model selection can be made either by means of Bayesian regularisation framework, comparative analysis of the generalisation errors (estimated with the support of, for example, cross-validation) or by means of other suited regularisation approaches.

c) The essence lies here in the evaluation framework. So, this study should test cross-patient generalisation by splitting data into training, validation and test sets all including data collected from different subjects. This should ideally be performed within the cross-validation framework.

Even if cross-patient differences are considerable, it would be relevant to exploit the data for unsupervised pre-training of deep architectures (so the data without annotations would be exploited).

d) The problem would be concerned with the curse of dimensionality – too many features with too few sample data points. One could approach this challenge through feature analysis/selection.

e) The recordings should be treated differently across a day if they differ within a day, which would imply that the daily average is not necessarily a representative measure of the situation that evolves throughout the day (likewise, it may also imply that the annotation as a daily average does not account for the evolution/change of mood throughout a day).

If we stick to labelling mood once a day, we should attempt to understand if the daily mood score (a single label mentioned) can be explained by the sequence of measurement patterns, and consider the application of time-delayed feedforward or recurrent neural networks (e.g., LSTM).

Question 9 (5p)

Averaging helps in reducing the variance in the statistical learning sense (one could say less dependency on the selection of the training data). (1p)

The less correlated the errors of individual weak learners are the more variance is reduced, which implies that the network should be "independent" (rather provide orthogonal/ uncorrelated outputs; have diverse properties with respect to the input), which can be achieved by, e.g., training individual (weak) learners on different subsets of the training data. We also tend to choose individual networks with low bias since the variance will be taken care of by the ensemble. (1p)

As representatives of ensemble learning approaches one could use bagging, boosting or mixture of experts (which display dependence on input). (2p, 1p/algorithm)

A committee of networks produces model with greater generalisation capability due to the improved bias-variance balance (for this reason, we tend to choose individual networks with low bias as the variance is taken care of by the ensemble). Interpretability can be enhanced for mixture of experts where the input decides about the selection of networks to determine the output. (1p, *it is ok if they do not comment on the interpretability*).

Question 10 (6p)

a) It is a classification problem (in special circumstances it could be seen as a regression problem). (1p)

A multi-layer perceptron (feed-forward architecture with hidden layers) can offer a straightforward solution. The number of inputs is determined by the number of attributes, i.e. 122000 (if there was no preprocessing resulting in the dimensionality reduction). The number of outputs will correspond to the number of states, so it is the number of error states plus one. (1p)

Training is done by means of gradient descent based optimisation, e.g. backprop. (1p)

b) Self-organising map (SOM) / Kohonen network lends itself to addressing this mapping problem where topological properties of the input space should be preserved. (1p)

The size of the input space is determined by the number of attributes and the nodes could be arranged in a 2D rectangular grid (corresponding to the layout of the exhibition hall) with neighbourhood conveying the intuitive notion of proximity between the locations of exhibitors. (1p)

For training, a competitive-collaborative SOM algorithm should be used. (1p)

Question 11 (4p)

Reservoir is a recurrent neural network. (0.5p)

To obtain better separation between states reservoir should be a large network with sparse (1-20%) connectivity (very often it is designed as a network with the small world connectivity pattern) and small spectral radius (less than 3) promoting stability. (0.5p, it is enough to mention sparseness or any two characteristics)

Training echo state networks (ESNs) is fast since it is only concerned with training output weights, which is a linear problem so we end up with a least mean square problem. (1p)

The fundamental differences in solving the problem of sequence learning between ESNs and typical (vanilla) recurrent neural networks (RNN) or long short-term memory (LSTM) networks lies in learning/handling time dependencies. In RNNs we adapt "nonlinear" network parameters through training (backpropagation through time) while in ESNs these nonlinear parameters involved in recurrent processing are fixed and only the linear read-out weights are adjusted with linear methods. (1p)

LSTMs help in addressing the problem of vanishing gradients that vanilla type of RNNs suffer from, which allows LSTM to handle/learn longer sequences (longer time dependencies). In this regard, gating units in LSTM cells play a critical role. (1p)

Question 12 (6p)

Data representation is a way of *representing* (*coding*) salient information, key characteristics (features) of the data describing objects. (0.5p)

Learning representations implies a capability to *automatically* discover data representations/characteristics (features) suitable for (pattern recognition) tasks at hand as part of the learning process unlike hand (manual) feature engineering as input to machine learning algorithms/neural networks prior to any learning. (1.0p)

Deep networks help distribute the task of learning representations across layers. In this regard, *layer-wise pretraining* particularly supports the process of *extracting hierarchies of features*. In shallow networks the input has to already be represented in terms of features (*hand engineered* beforehand). Learning representations in deep networks enhances their generalisation and functionality (deep networks provide more room for representing *hierarchy of explanatory factors*, potentially *shared across many tasks*). (0.5p)

Extracting suitable data representations in deep belief nets (DBN) is implemented by greedy training layer-by-layer restricted Boltzmann machines (RBMs). *Contrastive divergence* algorithm is used for that purpose. In the iterative process of mapping a visible representation to a hidden representation and vice versa (by means of Gibbs sampling), the reconstruction error is minimised and thus hidden layer converges to convey a probability distribution of the data over latent variables ("hidden" explanatory factors for the original input data/visible part of "real" data). (1.0p)

Stacked autoencoders are implemented by greedy layer-by-layer training of autoencoders (preserving hidden layers, without the output), relying on gradient descent optimisation. To avoid trivial mappings in overcomplete autoencoders (with the higher dimensionality of the hidden layer than that of the input/output) we use regularisation. A practical approach is to use denoising autoencoders (data manually corrupted with noise). (0.5p)

In *undercomplete autoencoders* regularisation is inherent due to the lower dimensionality of the hidden layer relative the input, which implies that the mapping problem is usually *underdetermined* by nature (non-trivial compression). (0.5p)

Information is distributed across many units that account for information about features that are not mutually exclusive. So, each unit represents more than one object, it represents one of its features that can be shared with other objects. This overlap between representations determines similarity between objects, which facilitates generalisation. This approach is in the spirit of hierarchical representations of features across layers in deep architectures. (1.0p)

In a local representational scheme, each object is represented by a single representational element (independently), e.g. *"one-hot"* representation. (0.5p)

Sparse distributed representations are distributed representations of typically large dimension with only a few unites being active at the same time (there are a lot of quiet units in the representation of each object). (0.5p)