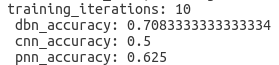
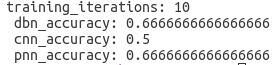
1. Discussion of architectures

* I’ll work with structured data, represented as a CSV file. the Breast Cancer Coimbra Data Set. Clinical features were observed or measured for 64 patients with breast cancer and 52 healthy controls. 116 number of instances total. There are 10 predictors (features in our case), all quantitative, and a binary dependent variable (label in our case), indicating the presence or absence of breast cancer.
* The predictors are anthropometric data and parameters which can be gathered in routine blood analysis. Prediction models based on these predictors, if accurate, can potentially be used as a biomarker of breast cancer.
* There are many use cases where the amount of training data available is restricted. Our case is no different. In our case our data is labeled, so Deep Belief Networks might be suitable for our case. We have limited training data, so Multilayer perceptron feedforward artificial neural network model should show comparable accuracy results. Convolutional Neural Networks model is not suitable in our case.
* Perceptrons (computational models of a single neuron) is also called feed-forward neural network, training perceptrons usually requires back-propagation, giving the network dictionary of inputs and outputs. Inputs are sent into the neuron, processed, and result in an output. The error being back propagated is some variation of the difference between the input and the output. Given that the network has enough hidden neurons, it can always model the relationship between the input and output. Practically their use is a lot more limited but they are popularly combined with other networks to form new networks. Disadvantages of perceptron: If you choose features by hand and you have enough features, you can do whatever you want. For binary input vectors, we can have a separate feature unit for each of the exponentially many binary vectors and so we can make any possible discrimination on binary input vectors. But once the hand-coded features have been determined, there are very strong limitations on a perceptron learning. Convolutional Neural Networks are different from most other networks. They are primarily used for image processing but can also be used for other types of input such as as audio. So this neural network model won’t show great results in our case. Convolutional network layers tend to shrink as they become deeper, mostly by easily divisible factors of the input.  Deep Belief Networks model use technique known as greedy training, where greedy means making locally optimal solutions to get to a decent but possibly not optimal answer. Using belief network, we get to observe some of the variables. We avoid the inference problem: Infer the states of the unobserved variables, and The learning problem: Adjust the interactions between variables to make the network more likely to generate the training data. Deep Belief Networks can be trained using both contrastive divergence and back-propagation and learn to represent the data as a model. Once trained/converged to a stable state through unsupervised learning, the model can be used to generate new data. If trained with contrastive divergence, it can even classify existing data because the neurons have been taught to look for different features.
* I’m going to use Deep Belief Networks architecture because of labeled data. And we have low number of data, so bad learning time scale problem, which means it is very slow in networks with multiple hidden layers won’t be a problem.

2. Creation and aplication of a neural network

* Data contains 9 inputs (features): Age, BMI, Glucose, Insulin, HOMA, Leptin, Adiponectin, Resistin, MCP.1 and one output: Classification. I preprocessed data using sklearn library preprocessing module StandardScaler class fit\_transform function used for labeled supervised transformation. Another preprocessing method I used Is simply dividing by 16 and transforming value to float32 type.
* Boltzmann machine learning algorithm (restricted Boltzmann machine algorithm) is used in deep belief network pretraining in order to train stack by stack with high precision. For training back-propagation is used. For CNN used back propagation in a feedforward net with hidden layers. For both I had to initialize hidden layers with, input and output layers, training steps, learning rate features (input) data ant label (output) data which I splitted into training and testing set. Nh=Ns/(α∗(Ni+No)) Ni = number of input neurons. No = number of output neurons. Ns = number of samples in training data set. α = an arbitrary scaling factor usually 2-10. I used this formule in order to calculate the number of neurons I need in my hidden layers. I had to test α in order to check how much is enough. My decision in neurons quantity determined by 3 recomendations:

  
Illustration 2: results with sklearn StandartScaler

  
Illustration 1: results by using type transform preprocessing

1. The number of hidden neurons should be between the size of the input layer and the size of the output layer.
2. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
3. The number of hidden neurons should be less than twice the size of the input layer.

I Used 2 hidden layers. Batch size I used for training is 256 for speed purposes.

|  |  |
| --- | --- |
| none | Only capable of representing linear separable functions or decisions. |
| 1 | Can approximate any function that contains a continuous mapping from one finite space to another. |
| 2 | Can represent an arbitrary decision boundary to arbitrary accuracy with rational activation functions and can approximate any smooth mapping to any accuracy. |
| >2 | Additional layers can learn complex representations (sort of automatic feature engineering) for layer layers. |

3. Results and evaluation

Poor training results were observed with (data / 16).astype(np.float32) preprocessing function. Poor performance with low batch size and 3 hidden layers.



For convergence I had to train PNN 700 times, DBN 400 times and CNN 600 times. Futher training iterations examinations not proceeded in order to avoid overtraining because error increased. Few times during test runs DBN model achieved 93%. The upper bound on the number of hidden neurons that won't result in over-fitting was 10.

