Dealing with the open world classification problem using neural networks

Nikhil R. Pal

Indian Statistical Institute, Calcutta
Most classification problems are open world problems but treated as closed world (?).

**Closed world**: Test data come from the classes on which the classifier is trained- even in the closed world there is a problem!

**Sampling Window?**
Train an MLP (or any classifier) on data from some known classes.

A test data point comes from outside the “sampling window”.

What will the network/classifier do?

What should the network/classifier do?
Situation -2: Closed World

- Train an MLP or any classifier on data from some known classes.

Yes, this is possible

- A test data point from outside the “sampling window” gets classified to an “unrealistic” class!
- Is this OK?
- What should the network/classifier have done?
**Situation -3: Closed World**

- The statistical properties of one or more class change significantly with time (concept drift) causing a change in the sampling window.
- A test points comes from the drifted (but unknown) distribution.
- What should the network/classifier do?
- The classifier should not make any decision for the drifted data.
Situation -4: Open World

- A classifier is trained on four kinds of childhood cancer (Neuroblastoma, Rhabdomyosarcoma, Ewing Sarcoma, and non-Hodgkin lymphoma).
- A patient comes with Medulloblastoma (another SRBCT)
- What will the network do?
- What should the network do?

**Commonality**: Arise only when the test point is outside the sampling window of the training data.

Is there a common solution to all four?
Common solution?

- Possibly yes
- But, if the sampling window for the new class is not different – we have a problem!

Possible Solution

- No decision (Don’t know) if a test sample comes from “outside” the “sampling window”.

- We need to estimate the “sampling window” based on the training data.
So far no reference to Neural Networks

Why the problem is important for NNs!
Issues with NNs

- A wonderful tool, has many applications (forecasting, function approximation, cancer screening, satellite image analysis, bioinformatics ....), but there are issues that need serious attention.
- Primarily we focus on the multilayer perceptron.
- As such it cannot deal with the open-world nature of classification problems.
- Cannot say don’t know when it should.
- Lack of incremental learning ability.
- Lack of interpretability of network.

MLP is an example, the principle that we shall discuss can be used with RBF also
The Multilayered Perceptron (MLP)

• Layered feed-forward networks.
• The $n^{th}$ layer is fully connected with the $(n+1)^{th}$ layer.
• Input nodes are fan out nodes.
• Each of the other nodes computes a weighted sum of its inputs and applies a sigmoidal function.
For an MLP, the $i^{th}$ node of first hidden layer computes:

$$\text{Net}(x) = \sum_{j=1}^{p} W_{ij} x_j = w_i^T x$$

For Sigmoidal nodes:

$$\phi_i(x) = \frac{1}{1 + e^{-\text{Net}(x)}}$$

Assume

$$x = \begin{bmatrix} x_1 & x_2 & \ldots & x_p \end{bmatrix}^T$$

$$w_i = \begin{bmatrix} w_{1i} & w_{2i} & \ldots & w_{pi} \end{bmatrix}^T$$

$\# \text{ features}$
We focus on Classification problems- NOT function approximation type problems.

The response of an MLP for a data point which is away from its training sample may be very unreliable.

We consider a point classified to class $k$, If the output of the $k^{th}$ class is more than 0.8 and outputs from all other classes are less than 0. (Normally, this is NOT done)
Unwanted generalization – an example

Three-class training data
(100 points per class)

Generalization by ordinary MLP
(20 hidden nodes, white – no decision)

→ It does not mean that every time we train an MLP, we get such a picture, but it is not infrequent either – On the other hand, with fuzzy rules, it would be difficult to get such a picture.
Two class training data (500 points per class)

Generalization by ordinary MLP (20 hidden nodes gray – no decision)
Dish-shell training data (500 points per class)

Generalization by MLP (20 hidden Nodes; White/gray – no decision)
Propose two solutions

- A simple one with incremental learning as a by-product.

- An improved method backed by theory
Making an MLP a better decision maker:

Given

\[ X = \{x_1, x_2, \cdots, x_n\} \subset \mathcal{R}^p \]

\( X \) has \( k \) classes & \( X = \bigcup_{i=1}^{k} S_i; \ x_j \in S_i \)

is from class \( i \)

We want generalization on points in the “neighborhood” of a class as defined by the training data and beyond that it should NOT make any decision

Let

\[ \Omega \] be the smallest hypercube bounding \( X \) inflated by \( l\% \) (\( l = 5 \)) on all sides.
Generate points $Y_1$ within $\Omega$, but outside the ‘boundary’ of $S_1$

Generate points $Y_2$ within $\Omega$, but outside the ‘boundary’ of $S_2$
Let $Y_i$ be a set of points, generated uniformly within $\Omega$ but outside the “boundary” of $S_i$.

Construct $k$ training sets

$T_i = S_i \cup Y_i$, $i = 1, 2, \ldots, k$;

$S_i$ with label 1 and

$Y_i$ with label 0

We train an MLP $M_i$ with $T_i$; $i = 1, 2, \ldots, k$

$\Rightarrow$ each $M_i$ learns a 2-class problem

Merge them by a Simple Merge to get a Composite network
Simple Marge
Given a test input $x \in \mathbb{R}^p$, the composite net will produce a $k$-dimensional output. Then for each $x$ at most one subnet is expected to produce a high response. ➔ either the class label of $x$ is Unambiguous Or no decision should be made.

If NO overlap Between the training data from different classes
If the training data from classes \( i \) and \( j \) overlap, and if \( x \) is from the overlapped region, then output of both classes \( i \) and \( j \) will be high. In such a case we should not assign \( x \) to any one of \( i \) and \( j \). A better choice would be to say that \( x \) can be in either of the 2 classes—A signal for overlapped region.
With conventional MLP

if there are overlapped classes, it should (will surely) produce misclassifications - training may become unstable.

If an MLP learns overlapped classes without error, then it is over-fitting on the data.

We cannot interpret the outputs to detect -whether a test point belongs to an overlap area.
Proposed network

• Obtains multiple class labels for data points in areas of overlap

• Also if the response for NO class is significantly high, no decision is made \rightarrow the Don’t know case

• It can also generate soft decisions in appropriate situations
But, our success depends on the concept of boundary!

A point $x$ is outside the boundary of the pattern class $c$, if the nearest neighbor of $x$ in $S_c$ is at a distance greater than $\alpha \cdot \delta_{avg}^c$. Otherwise, it is inside the boundary of $S_c$.

$\delta_{avg}^c = \text{average edge length of a Minimal Spanning Tree spanning points in class } S_c$

$\alpha \geq 1$ is a predefined constant.
If $x > \alpha \cdot \delta_{avg}^C$ then outside the boundary of $S_1$. 
The tightness of the boundary depends on the value of $\alpha$.

A smaller value of $\alpha$ will yield a tighter boundary.

If $S_i$ forms a nice cluster,

then $\delta_{avg}^i$ gives a good idea about the interpoint distances.
Even if $S_i$ forms a few well separated clusters, & if the density of points in each cluster is almost the same then also this will be effective.

$\delta^i_{avg}$ is not affected much by the number of clusters & the “distance” between the clusters in $S_i$, for large $|S_i|$. 

Do we really need to find the clusters?
Scatterplot of Square
Points generated outside the boundary of Square for different $\alpha$
Points generated outside the boundary of Dish-shell for different $\alpha$
### Experimental Results

<table>
<thead>
<tr>
<th>Name</th>
<th>Features</th>
<th>Classes</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dish-Shell</td>
<td>2</td>
<td>2</td>
<td>1000 (trn)</td>
</tr>
<tr>
<td>3-D Elongated</td>
<td>3</td>
<td>2</td>
<td>1000 (trn)</td>
</tr>
<tr>
<td>Scattered</td>
<td>2</td>
<td>3</td>
<td>600 (init) + 400 (incr)</td>
</tr>
</tbody>
</table>
Input

Generalization by the Proposed method – Gray represents don’t know

Generalization by Ordinary MLP
Scatterplot of $X_{init}$

Generalization on $X_{init}$

Black – no decision
Features 1-2

Input

Features 2-3

Features 1-3

Elongated Data
Features 1-2

Results

Features 2-3

Tested on $\Omega$ with 10,000 points randomly generated. White $\rightarrow$ NO Decision

Features 1-3
Problems with Higher Dimensional Data

Many points are to be generated outside the boundary to properly represent the structure of a class, so training of subnets becomes expensive.

A solution may be to generate fewer points and use points from other classes which lie outside the boundary of the class in question.
Initially we have a data set

We construct a network

we obtain a new a data set

\[ X_{\text{init}} : k \text{ classes} \]

\[ M_{\text{init}} \text{ with } X_{\text{init}} \]

\[ X_{\text{new}} : m \text{ classes} \]

\[ m = k + n \]

\[ X_{\text{new}}^O \Rightarrow \text{old } k \text{ classes and } X_{\text{new}}^N \Rightarrow \text{new } n \text{ classes} \]
Old data on k-classes

Old net

Simple merge

Train n subnets in the same manner

Compound Merge

DESIRED NET

Train at most k subnets in the same manner

Old data on k-classes

New data

on new n-classes

New data

on old k-classes

or

New data

Done

New data

Old data on k-classes

Incremental learning of MLPs
Compound merge of two trained MLPs

\[ M^l_{\text{init}} \quad \text{AND} \quad M^{i_j=l}_{\text{new}} \]

\[ M^l_{\text{new}} \quad \text{AND} \quad M^{i_j=l}_{\text{new}} \]
Scatterplot of $X_{\text{init}}$

Scatterplot of $X_{\text{init}} \cup X_{\text{new}}$
Generalization on $X_{init}$

Black – no decision

Generalization on $X_{init} \cup X_{new}$
Remarks

The philosophy may be used with other classifiers too.

It is not readable, but slightly more comprehensible than usual MLPs.

Better definition of boundary is required with some theoretical justification.

For high dimensional data, better methods to generate points from outside the boundary are required.

Train only one classifier instead of c classifiers.

Next we try to address some of these issues.
Training data \( \{x_1, x_2, \ldots, x_n, \ldots\} \); \( x_i \in \mathbb{R}^p \) are independent random samples from a probability space \((\mathbb{R}^p, \mathcal{B}^p, \mathbb{P})\); \( \mathbb{R}^p \) is the \( p \)-dimensional Euclidean space, \( \mathcal{B}^p \) is the associated Borel \( \sigma \)-field; \( \mathbb{P} \) is a probability measure, \( \mathbb{P} \) maps \( A \in \mathcal{B}^p \) to \([0,1]\)

Denote \( X_n = \{x_1, x_2, \ldots, x_n\} \)

- The sampling window of the data is the smallest set outside which \( \mathbb{P} \) has zero mass.
- So our problem is to estimate the support of the probability measure \( \mathbb{P} \) using the training data.
Estimation of support (Sampling window)

\[ D(X_n) \] is the smallest hyperbox (inflated a little) containing \( X_n \). The data are scaled to limit in \([0,1]\).

The smallest hyper-box covering the data points (2-D illustration)
Estimation of support (Sampling window)

Partition of the hyper-box

Denote $D_{m,i}(X_n) = \text{i}^{th}$ Small hyperbox

Divide $D(X_m)$ into hyperboxes of sides $1/m$

We define the sequence of sets $\{ D_{m,i}(X_n); i \geq 1 \}$

$m$ and $n$ are finite, so for large $i$ assume $D_{m,i}(X_n) = \emptyset$
Estimation of support (Sampling window)

Mark a hyper-box if it contains at least a data point

Union of the marked boxes is taken as an estimate of the support based on $n$ samples.

For $A \subseteq \mathbb{R}^p$

$$T(A; X_n) = \begin{cases} A & \text{if } x_i \in A \text{ for some } x_i \in X_n \\ \phi & \text{otherwise} \end{cases}$$
Estimation of support (Sampling window)

So our estimate is
\[
\bigcup_i T(D_{m,i}(X_n); X_n)
\]

If the true support is \( C \), the error in estimate is
\[
\bigcup_i T(D_{m,i}(X_n); X_n) \Delta C
\]

\( \Delta = \) symmetric set difference operator

For \( A \subseteq \mathbb{R}^p \)

\[
T(A; X_n) = \begin{cases} 
A & \text{if } x_i \in A \text{ for some } x_i \in X_n \\
\phi & \text{otherwise}
\end{cases}
\]
When $n$ is large the worst case limiting error of our estimate of $C$ is negligible (in an almost sure sense), if we take $m$ large enough.

Under some assumptions it can be proved that

$$\lim_{m \to \infty} P \left( \lim_{n} \sup_i T \left( D_{m,i}(X_n); X_n \right) \Delta C \right) = 0$$

For a Rigorous Proof see: B Karmakar, N R Pal, How to make a neural network say “Don’t know”, *Information Sciences*, Vol 430-431, pp 444-466, 2018
How to use the estimate of the sampling window?

- Extend the training data uniformly to non-informative region; i.e., in the complement set of the support (Don’t know class), $X_c$
- Train a network with $X_n$ and $X_c$; a (c+1) class problem.
- But even for a reasonably high dimension, $X_c$ could be very large.
  - Too many samples are needed for a faithful representation
  - Support regularization
- Minimize input sensitivity in the complement region.
- Smoothness in the “don’t know” class WRT input variation.
Support regularization and learning algorithm

- Statistical sensitivity $S(W,X_c)\text{**}$

$$S(W,X_c) = \lim_{\sigma \to 0} \frac{\sqrt{\text{Var} (\nabla \text{Output})}}{\sigma}$$

- $\nabla \text{Output}$ is the error in the output due to perturbation of input;
  $\sigma$ is the standard deviation of each component of the input perturbation; $\text{Var}$ is the sample variance function.


The learning objective is:

$$E(W,X_n,X_c) = L(W,X_n,X_c) + \lambda S(W,X_c); \quad \lambda > 0$$
Support regularization and learning algorithm

- There are several ways to extend to multiclass outputs.
- In case of additive (also for multiplicative) noise for an MLP, $S(W, X_c)$ takes simple form which is independent of $\sigma$.
- We considered additive noise and in this case $S(W, X_c)$ becomes norm of a vector whose elements involve $W$ and $X_c$.

The learning objective is:

$$E(W, X_n, X_c) = L(W, X_n, X_c) + \lambda S(W, X_c); \; \lambda > 0$$
Generation of $X_c$

- **Step 1:** Choose $n_1$ and $m$.
- **Step 2:** $N ← 0$, $X_c = \emptyset$ (empty set)
- **Step 3:** Repeat while ($N \neq n_1$)
  - 3.1: Randomly choose one of the hyper-boxes $D_{m,i}(X_n)$
  - 3.2: If, no $(d/2 +1)$ or larger section the hyper-box contains the corresponding sections (components) of $x_1, x_2, \ldots, x_n$, choose a random sample ($x$) from this hyper-box.
    
    Set $N ← N + 1$
    $X_c = X_c \cup x$
    
    Else go to 3:1.

Suppose $d=5$ and we select $D_{m,i}(X_n)$ with boundaries 0.2-0.3; 0.4-0.5; 0.1-0.2; 0.6-0.7; and 0.1-0.2. No training point falls exactly in $D_{m,i}(X_n)$, but $\exists$ a training point (.25, .42, .15, .92, .32) - in this case we discard $D_{m,i}(X_n)$ as 3 components (more than $d/2$) fall in the box. $\Rightarrow$ we discard this box!
Choice of m and Decision making

- Find a minimum spanning tree for each class in the training set.
- Choose box size $1/m = \text{the maximum of all edges of the spanning trees}$.
- Data from a class form a set of clusters $\Rightarrow$ Compute MST on each cluster.
- Choose box size $= \text{the maximum of all edges}$.
- Empirical evidence shows it is a good choice.

Decision Making

- There are $k$ classes $\Rightarrow$ output vector $o_x \in R^{k+1}$, $(k+1)$th class is the Don’t know class.
- Decide class $c$, if $o_{x,c} =\max_i \{o_{x,i}\}$ and $c \neq (k + 1)$; $o_{x,c} > 0.8$, $o_{x,i} < 0.2 \ \forall i \neq c$; otherwise assign the don’t know class.
Decision making

- Given an $x$ we can make two types of classification: using just the maximum support and the threshold based one.
- We refer to the first one as the "region classifier" and the second one as the "decision region".
Results: Two Synthetic, some UCI data sets & many more

3-class data set
1000 points;

Annular data set
1000 points

$m=1/0.2; \lambda=0.01$ (adhoc choice)
Only $n_1=n/2$ points are generated from the complement world; MLP with 10 hidden nodes
Results – Synthetic 3-class data

Top: Generalization by the back-propagation algorithm (a typical run)

Left: Generalization by the proposed algorithm (same initialization)

→ White don’t know class
→ Not very good, why?
→ Large box size?

m=1/0.2 (adhoc choice)
Optimal choice (MST): $m = 1/0.0056$ (its nearest integer, 179, to be precise).

Larger box size, $m=50$

Smaller box size, $m=300$

All results are with the same initial condition
Results – Annular data

Optimal choice (MST): $m = 1/0:0016$

Threshold based decision on MLP

All results are generated with the same initial condition
Results – Wine data

Proposed method (with thresholds)

MLP with c+1 classes (use of max)

Optimal $m = 1/0.0016$

Why?

Fraction of points with minimum distance from the training data in $(x-\delta, x+\delta)$ that is classified as don’t know

Generate samples uniformly from the entire hyper-box; analyze $d(x,x_k)$, where $x$ is assigned the don’t know class and $x_k$ is the closest point from training set.

Three classes: 59, 71 and 48 observations in 13-D

Complement region: just 89 points in 13–D! Very challenging!

89 points in 13–D may not be enough to represent the complement region! So take $n_1=n$
You have already seen what we did!
Did we solve the problem?
No, for very high dimensional data, we still have a computational bottleneck
Choice of $\lambda$ is an issue.
Choice of $m$ also needs further investigation.
Use of possibly better regularizer?
Can extreme value theorem help?
Thank you