Weekly Report 11

Utkarsh Patel
18EC30048

Topics Covered

1 Perceptron Modelling a Neuron

If we have a training instance $\boldsymbol{x} = (x_1, x_2, \dots, x_n)$, then we can use perceptron modelling to find $z = \boldsymbol{w}^T\boldsymbol{x} + b$, and use non-linearity $a = g(z)$ for classification purpose. The most common non-linearities used are: $ReLU(\cdot)$, $\tanh(\cdot)$ and $sigmoid$ function.

2 Artificial Neural Network

ANN is a feed-forward fully connected network of perceptrons or what we call neurons. It may have several hidden layers. There exists no feedback or loop in the network.

3 Forward Propagation in ANN

Let k^{th} layer has $n^{[k]}$ neurons. So, if we are considering i^{th} layer, each of its $n^{[i]}$ neurons are fully-connected to $n^{[i-1]}$ neurons of previous layer. Therefore, for $j = 1, 2, \dots, n^{[i]}$, we can model each neuron with a logit unit given as

$$z_j^{[i]} = \boldsymbol{w}_j^{[i]^T}\boldsymbol{a}^{[i-1]} + b_j^{[i]}$$
$$a_j^{[i]} = g\left(z_j^{[i]}\right)$$

Hence, the output of i^{th} layer is given as $\boldsymbol{a}^{[i]} = \left(a_1^{[i]}, a_2^{[i]}, \dots, a_{n^{[i]}}^{[i]}\right)^T$. Usually, in this notation, the input feature vector $\boldsymbol{x}$ is denoted by $\boldsymbol{a}^{[0]}$. Hence, using vectorized representation, we have,

$$\boldsymbol{z}^{[i]} = \boldsymbol{W}^{[i]}\boldsymbol{a}^{[i-1]} + \boldsymbol{b}^{[i]}$$
$$\boldsymbol{a}^{[i]} = g(\boldsymbol{z}^{[i]})$$

Here, $\boldsymbol{z}^{[i]}, \boldsymbol{a}^{[i]}, \boldsymbol{b}^{[i]} \in \mathbb{R}^{n^{[i]}}, \boldsymbol{a}^{[i-1]} \in \mathbb{R}^{n^{[i-1]}}$ and $\boldsymbol{W}^{[i]} \in \mathbb{R}^{n^{[i]} \times n^{[i-1]}}$.

4 Optimization

ANNs are used both in classification and regression tasks. The error function, also called loss function in this domain, is *mean-squared error* for regression and *cross-entropy* for classification. The weights and biases of all the layers are updated so as to make this loss function minimal. Gradient-based optimization is used in ANNs and the gradients are propagated using chain rule.

5 Back Propagation in ANN

Let the ANN of interest has l layers, with input being layer 0 and output being layer l. The loss function L, which is a function of weights and biases of all the layers is defined directly from $\boldsymbol{a}^{[l]}$, the output of layer l. In back propagation, we use $\partial\vartheta$ to represent $\frac{\partial L}{\partial \vartheta}$. Therefore, $\partial\boldsymbol{a}^{[l]}$ can be computed easily. This gradient is transferred through the layers, from output layer to input layer, using chain rule of differentiation. Consider that we computed $\partial\boldsymbol{a}^{[i]}$ for i^{th} layer, then, to compute $\partial\boldsymbol{a}^{[i-1]}$ we have,

$$\partial\boldsymbol{z}^{[i]} = \partial\boldsymbol{a}^{[i]} \cdot g^{[i]'}(\boldsymbol{z}^{[i]})$$
$$\partial\boldsymbol{W}^{[i]} = \partial\boldsymbol{z}^{[i]} \cdot \boldsymbol{a}^{[i-1]^T}$$
$$\partial\boldsymbol{b}^{[i]} = \partial\boldsymbol{z}^{[i]}$$
$$\partial\boldsymbol{a}^{[i-1]} = \boldsymbol{W}^{[l]^T} \cdot \partial\boldsymbol{z}^{[i]}$$

During this, the weights and biases of layer i are updated as follows

$$\boldsymbol{W}^{[i]} = \boldsymbol{W}^{[i]} - \eta\partial\boldsymbol{W}^{[i]}$$
$$\boldsymbol{b}^{[i]} = \boldsymbol{b}^{[i]} - \eta\partial\boldsymbol{b}^{[i]}$$

6 ANN Training

All the weights and biases are initialized randomly. However, how they are initialized have a significant effect on the training of the model. The weights and biases are updated after the loss function is computed over entire training sample, or over a batch, or for each sample depending upon whether we are using *gradient descent, batch gradient descent* or *stochastic gradient descent*. The process is continued till convergence is achieved.

7 Improving Convergence

The *gradient descent* achieves the optimal minima, however, as the weights are updated after the loss function is computed over entire training samples, it is very slow in practice. Therefore, we use *batch gradient descent* or *stochastic gradient descent* where weight updates are quite frequent. However, there exists a trade-off between accuracy and time. The latter techniques are quite noisy as they consider only a small part of training sample while updating weights, and hence they don't converge really and may oscillate around the minima indefinitely, or may diverge also. To counter such noise, we use momentum while using doing *batch* or *stochastic gradient descent*, where the weight update is not abrupt but a cumulation of previous weight updates and current descent. Hence, the noise is reduced to a large extent.

$$\Delta\boldsymbol{W}^{[i](t)} = \alpha\Delta\boldsymbol{W}^{[i](t-1)} + (1-\alpha)\left(-\eta\partial\boldsymbol{W}^{[i]}\right), 0 \le \alpha \le 1$$
$$\boldsymbol{W}^{[i](t)} = \boldsymbol{W}^{[i](t-1)} + \Delta\boldsymbol{W}^{[i](t)}$$

8 Clustering

A *cluster* is defined as a collection of objects similar to each other, but different from objects belonging to other clusters. They differ from *classes* in respect that they are a group with *loosely* defined similarity within its objects, however, in *classes*, objects are grouped by their well-defined characteristics. Each *cluster* has the potential to form a *class*. Clustering is useful if we want to get representatives for homogenous groups which leads to data redundancy, or detecting outliers.

19 K-means Clustering

Given m instances $\boldsymbol{x}_i \in \mathbb{R}^p$, we want to compute $K -$ partitions of space $\mathbb{R}^p$, such that the sum of squared distance between a data point and centre of its respective partition is minimized.

$$L = \sum_{k=1}^{K}\sum_{x \in C_k} \|x - \mu_k\|^2$$

where $\mu_k = \frac{1}{|C_k|}\sum_{x \in C_k}\boldsymbol{x}$ is the centre of k^{th} partition. Given m data points, number of ways we can partition it into K cluster is of order $\Theta\left(\frac{K^m}{K!}\right)$, thus being exponential with number of instances. Therefore, finding the best partition of given set of points is an NP-hard problem. However, we can use heuristics to design a greedy approach that could get us close-to-perfect partition. The Lloyd's algorithm (Batch K means) uses such an approach.

BATCH-K-MEANS

```
1.  Randomly initialize position of $K$ centres.
2.  Loop till position of centres don't change
        • Assign data points to cluster whose center is closest to it
        • Calculate position of $K$ new centres obtained.
```

Through this algorithm is fast, a better convergence is not guaranteed. A more conservative approach would be to update the centres after every data point is moved for one cluster to the other (if the overall loss reduces). If so, the convergence is guaranteed at quadratic rate and the entire process is linear in m, p and K. The disadvantage of this approach is that this algorithm only detects compact, hyper-spherical clusters, is sensitive to noise and outliers, may struck in local minima (highly sensitive to initialization of centres). Also, there is no estimate of K, the optimal number of clusters required. There exists an algorithm called $K -$means++ which chooses the initial centres statistically.

K-MEANS+ +

```
1.  Choose first centre $c_1$ randomly.
2.  The $i^{th}$ centre is choosen with the probability proportional to square of distance
    from previously slected $i-1$ centres
```

$$p(x') = \frac{\sum_{j=1}^{i-1}\left\|x' - c_j\right\|^2}{\sum_{t=1}^{m}\sum_{j=1}^{i-1}\left\|x^{(t)} - c_j\right\|^2}$$

10 Estimating K in K-mean Clustering

We use cluster validity indices and stability check methods to determine optimal value of K. Cluster validity indices involves external indices and internal indices. External indices use a reference partitioning information (e.g. class labels). Internal indices involve analyzing variance distribution structure of clusters. Stability check methods is based on the assumption that repeated clustering should result in similar partitioning for appropriate K.

WANG-ALGORITHM

```
1.  Permute inputs $c$ times.
2.  Each time divide the inputs into three parts $S_1, S_2$ and $S_3$, with $|S_1| = |S_2|$.
3.  Perform $K -$ means on $S_1$ and $S_2$, and test on $S_3$ to find the cluster numbers.
4.  Compute number of disagreement and take average over $c$ observations.
```

Bayesian inference and parametric methods can also be applied to achieve greater performance.

11 Hierarchical Clustering

It is a non-probabilistic approach and builds hierarchy of groups in bottom-up manner. Assuming that we have m instances, m clusters are formed each with one component. A graph G is initialized which contains a vertex for each cluster. While there exists more than one cluster, the algorithm picks two closest clusters say C_1 and C_2 using a distance look-up table. These clusters are merged to obtain a cluster C with $|C_1| + |C_2|$ elements. Distance of clusters with respect to cluster C_1 and C_2 are deleted from the look-up table. Distance of C is computed from every other cluster. These distances are appended to the look-up table. A new vertex C is added to graph G, whose children are C_1 and C_2. After the execution of algorithm, we get a hierarchical relationship between cluster in form a DAG. There exist several implementations of this algorithm depending on how distance between clusters is defined.

12 Corrupted Clique Problem

Given a graph G, we need to find minimum number of edges to be added/removed so as to transform this graph into a clique graph. It is an NP-hard problem; however, some heuristics exist to solve it approximately. The two popular algorithms are: PCC and CAST.

PARALLEL-CLASSIFICATION-WITH-CORES

```
1.  Let $S$ be the set of feature vectors, $G$ be the distance graph and $K$ be number of clusters.
2.  Randomly select subsets $S'$ and $S''$ such that $S' \cap S'' = \emptyset$, such that $|S'| = \Theta\left(\lg\left(\lg\left(|S|\right)\right)\right)$ and
    $|S''| = \Theta\left(\lg\left(|S|\right)\right)$.
3.  For all $K$ partition in $S'$
        • Obtain extended partition in $S$ through two stages of extensions.
        • Choose the one which has minimum score.
```

CAST-ALGORITHM

```
1.  Let $S$ be set of feature vectors, $G$ be the distance graph and $\theta$ be the distance threshold.
2.  $P \coloneqq \emptyset$
3.  Loop till $S$ is not empty
        • Let $v$ be the vertex of maximal degree in the graph $G$.
        • $C \coloneqq \{v\}$
        • Loop till a close feature $u$ not in $C$ or a distant feature $u$ in $C$ exists
              o Find nearest close feature $u$ not in $C$ and add it to $C$
              o Find the farthest distant feature $u$ in $C$
        • Add cluster $C$ to partition $P$.
        • $S \coloneqq S - C$
        • Remove vertices $v \in C$ from the graph $G$
4.  Return $P$.
```

Novel Ideas Out of Lesson

The way the weight matrices are initialized in MLP plays a crucial role in its training. For better perform, He initialization and Xavier initialization is used, and the sigmoid unit in the hidden layers are replaced with ReLU units for better performance. Apart from momentum, other techniques like RMSProp and Adam optimization in gradient descent are used for faster convergence.

Preparation of Quiz

Fair.