Hellinger Net: A Model for Imbalanced Learning

by

Tanujit Chakraborty

Postdoctoral Fellow, IIIT Delhi, India.

Statistical Consultant at Bajaj Finserv Ltd., Pune, India.

Outline of the talk

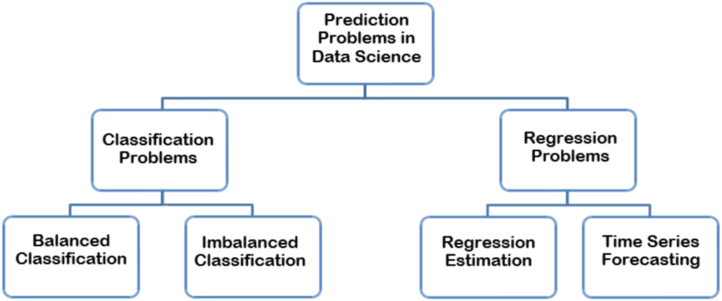
- Motivation
- Theoretical Background
- Hellinger Net Model
- Real-life Applications
- Simulation Study

Some Parts of the Talk is taken from:

T Chakraborty and AK Chakraborty. "Hellinger Net: A Hybrid Imbalance Learning Model to Improve Software Defect Prediction". **IEEE Transactions on Reliability**, 2020. **(Read Online)**

Motivation

"**Prediction** is very difficult, especially if it's about the future" - Niels Bohr, Father of Quantum Mechanics.



Some Popular Prediction Models

- Linear Regression (Galton, 1875).
- Linear Discriminant Analysis (R.A. Fisher, 1936).
- Logistic Regression (Berkson, JASA, 1944).
- k-Nearest Neighbor (Fix & Hodges, 1951).
- Parzen's Density Estimation (E Parzen, AMS, 1962)
- ARIMA Model (Box and Jenkins, 1970).
- Classification and Regression Tree (Breiman et al., 1984).
- Artificial Neural Network (Rumelhart et al., 1985).
- MARS (Friedman, 1991, Annals of Statistics).
- SVM (Cortes & Vapnik, Machine learning, 1995)
- Random forest (Breiman, 2001).
- Deep Convolutional Neural Nets (Krizhevsky, Sutskever, Hinton, NIPS 2012).
- GAN (Goodfellow et al., NIPS 2014).
- Deep Learning (LeCun, Bengio, Hinton, Nature 2015).
- Bayesian Deep Neural Network (Y. Gal, Islam, Zoubin, ICML 2017).

Need for Hybridization

- Statistical issue: It is often the case that the model space is too large to explore for limited training data, and that there may be several different models giving the same accuracy on the training data.
- Representation issue: In many learning tasks, the true unknown hypothesis could not be represented by any hypothesis in the hypothesis space. By hybridization, it may be possible to expand the space of representable functions.
- Computational issue: Many learning algorithms perform some kind of local search that may get stuck in local optima. Even if there are enough training data, it may still be challenging to find the best hypothesis.
- Data imbalance issue: Traditional classifiers assumes that the classes to be distinguished should have a comparable number of instances. Still, this assumption does not hold in real-world classification problems. The problem of learning from imbalanced data is a relatively new challenge that has attracted growing attention from both academia and industry.

Ensemble & Hybrid Models

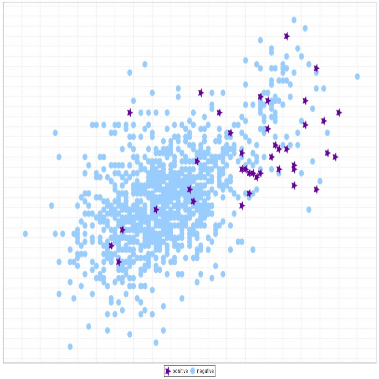
- Problem: Single models have the drawbacks of sticking to local minimum or over-fitting the data set, etc.
- Ensemble models are such where predictions of multiple models are combined together to build the final model.
- Examples: Bagging, Boosting, Stacking and Voting Method
- Caution: But ensembles don't always improve accuracy of the model but tends to increase the error of each individual base classifier.
- Hybrid models are such where more than one models are combined together.
- It overcomes the limitations of single models and reduce individual variance & bias, thus improve the performance of the model.
- Caution: To build a good ensemble classifier the base classifier needs to be simple, as accurate as possible, and distinct from the other classifier used.
- Desired: Interpretability, Less Complexity, Less Tuning Parameters, high accuracy.

Popular Hybrid Prediction Models

- Perceptron Trees (Utgoff, AAAI, 1988).
- Entropy Nets (Sethi, Proceeding of IEEE,1990).
- Neural trees (Sirat & Nadal, Network, 1990).
- Sparse Perceptron Trees (Jackson, Craven, NIPS, 1996).
- SVM Tree Model (Bennett et al., NIPS, 1998)
- Hybrid DT-ANN Model (Jerez-Aragones et al., 2003, AI in Medicine)
- Flexible Neural Tree (Chen et al., Neurocomputing, 2006)
- Hybrid DT-SVM Model (Sugumaran et al,, Mechanical Systems and Signal Processing, 2007).
- Hybrid CNN–SVM Classifier (Niu et al., PR, 2012).
- Hybrid DT model utilizing local SVM (Dejan et al., IJPR, 2013).
- Neural Decision Forests (Bulo, Kontschieder, CVPR, 2014).
- Deep Neural Decision Forests (Kontschieder, ICCV, 2015).
- Soft Decision Tree (Frosst, Hinton, Google AI, 2017).
- Deep Neural Decision Trees (Yang et al., ICML, 2018).
- Adaptive Neural Trees (Tanno et al. ICML, 2019).

Imbalanced Classification Problem

- Real-world data sets are usually skewed, in that many cases belong a larger class and fewer cases belong to a smaller yet usually more exciting class
- For example, consider a binary classification problem with the class distribution of 90 : 10. In this case, a straightforward method of guessing all instances to be positive class would achieve an accuracy of 90%.
- Learning from an imbalanced data set presents a tricky problem in which traditional learning algorithms perform poorly.
- Traditional classifiers usually aim to optimize the overall accuracy without considering the relative distribution of each class.



Sampling Techniques

- One way to deal with the imbalanced data problems is to modify the class distributions in the training data by applying sampling techniques to the data set
- Sampling technique either oversamples the minority class to match the size of the majority class or undersamples the majority class to match the size of the minority class.
- Synthetic minority oversampling technique (SMOTE) is among the most popular methods that oversamples the minority class by generating artificially interpolated data (Chawla et al., 2002, JAIR).
- TL (Tomek links) and ENN (edited nearest neighbor) are popular undersampling approaches (Batista et al., 2004, ACM SIGKDD).
- But these approaches have apparent deficiencies, such as undersampling majority instances may lose potentially useful information of the data set and oversampling increases the size of the training data set which may increase computational cost.
- To overcome these problems, “imbalanced data-oriented” algorithms are designed which can handle class imbalance without any modification to class distribution.

Effect of Class Imbalance on Decision Tree

Let X be attribute and Y be the response class. Here Y^+ denotes majority class, Y^- denotes minority class and n is the total number of instances. Also, let $X^{\geq} \longrightarrow Y^+$ and $X^{<} \longrightarrow Y^-$ be two rules generated by Classification Tree (CT). Table below shows the number of instances based on the rules created using CT.

Table: An example of notions of classification rules

class and attribute	$X^{\geq}$	$X^{<}$	sum of instances
Y^+	a	b	$a+b$
Y^-	c	d	$c+d$
sum of attributes	$a+c$	$b+d$	n

In the case of imbalanced data set the majority class is always much larger than the size of the minority class and thus we will always have $a+b >> c+d$. It is clear that the generation of rules based on confidence in CT is biased towards majority class.

Various measures, like information gain (IG), gini index (GI) and misclassification impurity (MI) expressed as a function of confidence, are used to decide which variable to split in the important feature selection stage, get affected by class imbalance.

Effect of Class Imbalance on Distance Measures

Table: An example of notions of classification rules

class and attribute	$X^{\geq}$	$X^{<}$	sum of instances
Y^{+}	a	b	$a+b$
Y^{-}	c	d	$c+d$
sum of attributes	$a+c$	$b+d$	n

Using Table, we compute the following:

$$P(Y^{+}/X^{\geq}) = \frac{a}{a+c} = \text{Confidence}(X^{\geq} \longrightarrow Y^{+})$$

For an imbalanced data set, Y^{+} will occur more frequently with $X^{\geq}$ & $X^{<}$ than to Y^{-}. So the concept of confidence is a fatal error in an imbalanced classification problem.

Entropy at node t is defined as:

$$\text{Entropy}(t) = -\sum_{j=1,2} P(j/t) log\big(P(j/t)\big)$$

Effect of Class Imbalance on Distance Measures

In binary classification, information gain for splitting a node t is defined as:

$$\text{IG} = \text{Entropy}(t) - \sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i) \tag{0.1}$$

where i represents one of the sub-nodes after splitting (assuming we have two sub nodes only), n_i is the number of instances in sub-node i and n is the total number of instances. The objective of classification using CT is to maximize IG which reduces to:

$$\text{Maximize}\left\{ - \sum_{i=1,2} \frac{n_i}{n} \text{Entropy}(i) \right\} \tag{0.2}$$

The maximization problem in eqn. (1.7) reduces to:

$$\begin{aligned} \text{Maximize}\Bigg\{ & \frac{n_1}{n}\Big[P(Y^+/X^{\geq})log\big(P(Y^+/X^{\geq})\big) + P(Y^-/X^{\geq})log\big(P(Y^-/X^{\geq})\big)\Big] \\ & + \frac{n_2}{n}\Big[P(Y^+/X^{<})log\big(P(Y^+/X^{<})\big) + P(Y^-/X^{<})log\big(P(Y^-/X^{<})\big)\Big]\Bigg\} \end{aligned} \tag{0.3}$$

The task of selecting the "best" set of features for node i are carried out by picking up the feature with maximum IG. As $P(Y^+/X^{\geq}) >> P(Y^-/X^{\geq})$, we face a problem while maximizing eqn. (0.3).

Theoretical Background: Consistency

- Statistical learning theory (SLT) studies mathematical foundations for machine learning models, originated in late 1960s.
- Basic concept of Consistency: A learning rule, when presented more and more training examples $\rightarrow$ the optimal solution.

Definition (Consistency)

Given an infinite sequence of training points $(X_i, Y_i)_{i \in N}$ with μ. For each $n \in N$, let f_n be a classifier for the first n training points. The learning algorithm is called consistent with respect to μ if the risk $R(f_n)$ converges to the risk $R(f_{Bayes})$, that is for all $\epsilon > 0$,

$$\mu(R(f_n) - R(f_{Bayes}) > \epsilon) \rightarrow 0 \text{ as } n \rightarrow \infty.$$

Definition (Universally Consistency)

The learning algorithm is called universally consistent if it is consistent for all probability distributions μ.

Theoretical Results on Decision Trees & Neural Networks

- Consistency of data driven histogram methods (Lugosi & Nobel, 1996, Annals of Statistics).
- A Fast, Bottom-Up Decision Tree Pruning Algorithm with Near-Optimal Generalization (Kearns, Mansour, ICML, 1998)
- Generalization Bounds for Decision Trees (Mansour et al., 2000, COLT).
- Consistency of Online Random Forest (Denil et al., 2013, ICML).
- Consistency of Random Forest (Scornet et al., 2015, Ann. Stat.).
- Strong Universal Consistency of FFNN Classifier (Lugosi & Zeger 1995, IEEE Information Theory).
- Approximation properties of ANN (Mhaskar, 1993, Advances in Computational Mathematics).
- Prediction Intervals for Artificial Neural Networks (Hwang, Ding, 1997, JASA)
- Provable approximation properties for DNN (Shaham et al., 2018, Applied & Computational Harmonic Analysis).
- On Deep Learning as a remedy for the curse of dimensionality (Bauer, Kohler, 2019, Ann. Stat.).

Consistency Results for FeedForward Neural Network Classifier

Theorem (Lugosi & Zeger, 1995, IEEE Information Theory)

Consider a neural network with one hidden layer with bounded output weight having k hidden neurons and let σ be a logistic squasher. Let $F_{n,k}$ be the class of neural networks defined as

$$F_{n,k} = \left\{ \sum_{i=1}^{k} c_i \sigma(a_i^T z + b_i) + c_0 : k \in \mathbb{N}, a_i \in \mathbb{R}^{dm}, b_i, c_i \in \mathbb{R}, \sum_{i=0}^{k} |c_i| \leq \beta_n \right\}$$

and let ψ_n be the function that minimizes the empirical L_1 error over $\psi_n \in F_{n,k}$. It can be shown that if k and β_n satisfy

$$k \to \infty, \quad \beta_n \to \infty, \quad \frac{k\beta_n^2 log(k\beta_n)}{n} \to 0$$

then the classification rule

$$g_n(z) = \begin{cases} 0, & \text{if } \psi_n(z) \leq 1/2. \\ 1, & \text{otherwise.} \end{cases} \tag{0.4}$$

is universally consistent.

For universal convergence, the class over which the minimization is performed has to be defined carefully. Above theorem shows that this may be achieved by neural networks with k nodes, in which the range of output weights $c_0, c_1, \ldots, c_k$ is restricted.

Hellinger Distance

Let (Θ, λ) denote a measurable space. Let us suppose that P and Q be two continuous distributions with respect to the parameter λ having the densities p and q in a continuous space Ω, respectively. Define HD as follows:

$$d_H(P,Q) = \sqrt{\int_\Omega (\sqrt{p} - \sqrt{q})^2 d\lambda} = \sqrt{2\left(1 - \int_\Omega \sqrt{pq} d\lambda\right)}$$

where $\int_\Omega \sqrt{pq} d\lambda$ is the Hellinger integral. It is noted that HD doesn't depend on the choice of the parameter λ.

For the application of HD as a decision tree criterion, the final formulation can be written as follows:

$$HD = d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{k} \left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}}\right)^2}, \tag{0.5}$$

where $|X_+|$ indicates the number of examples that belong to the majority class in training set and $|X_{+j}|$ is the subset of training set with the majority class and the value j for the feature X. The bigger the value of HD, the better is the discrimination between the features (Hellinger Distance Decision Tree, Chawla et al. 2008, ECML).

Hellinger Net : Basic idea

- Hellinger Net is composed of three basic steps:

  (a) Converting a DT into rules (HD is used as criterion);
  (b) Constructing a two hidden layered NN from the rules;
  (c) Training the MLP using gradient descent backpropagation (Rumelhart, Hinton (1988).

- In decision trees, the overfitting occurs when the size of the tree is too large compared to the number of training data.

- Instead of using pruning methods (removing child nodes), HN employs a backpropagation NN to give weights to nodes according to their significance.

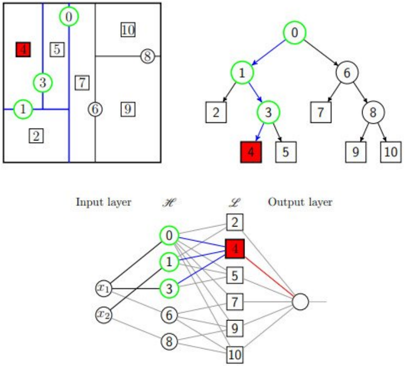
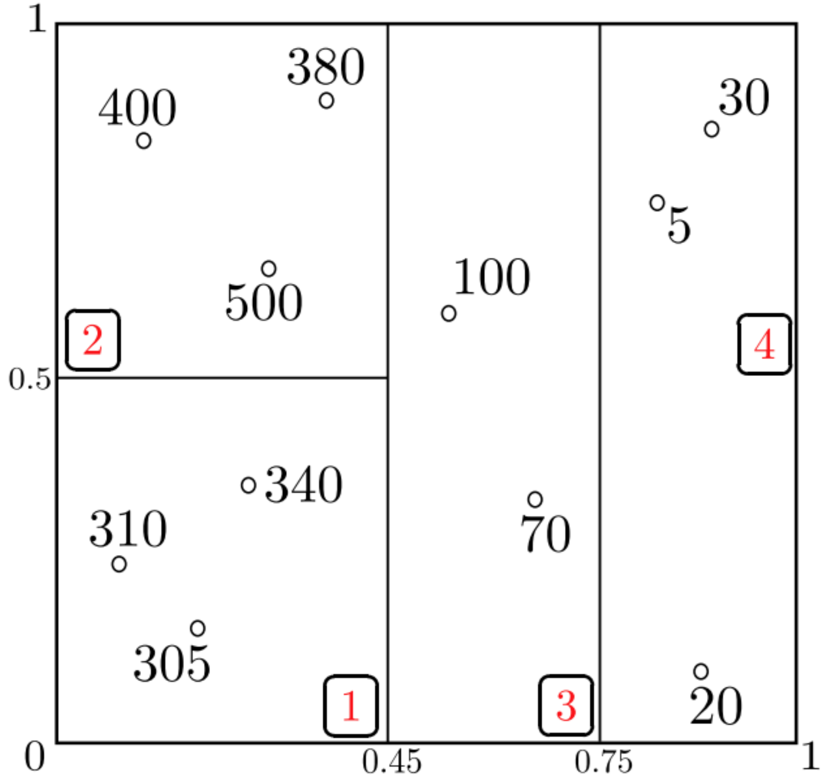
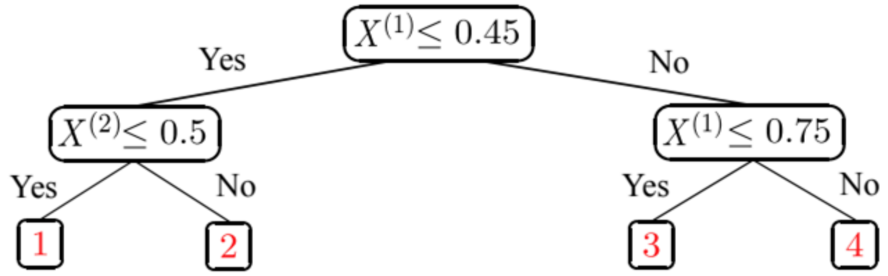


Fig: Graphical Representation of Hellinger Nets

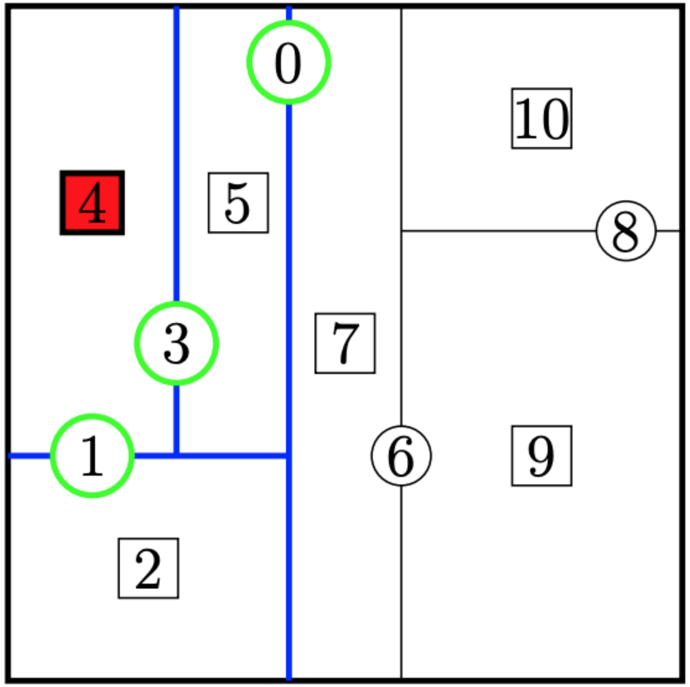
The idea of the this approach is inspired from the idea of Perceptron Trees [Paul E Utgoff, 1988, AAAI]

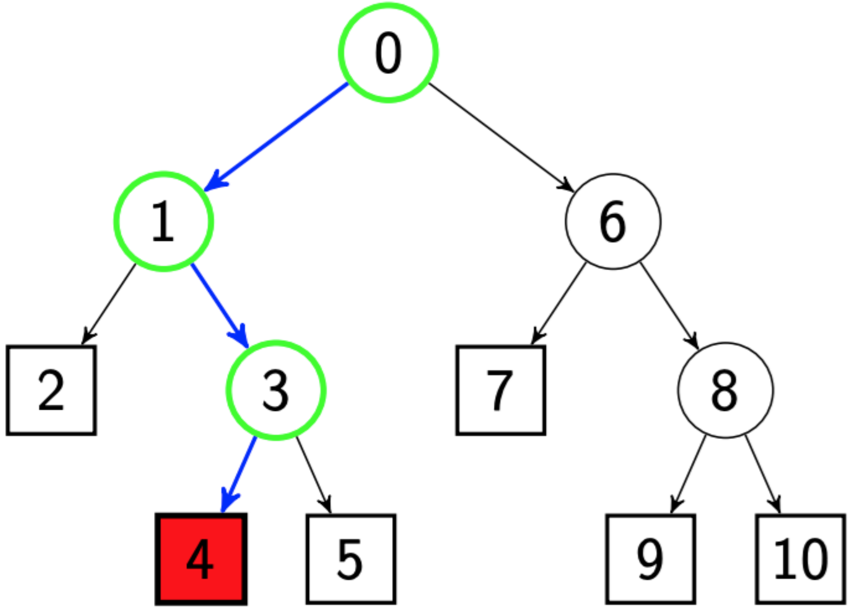
Partitioned Input Space & HDDT 1 / 2





Partitioned Input Space & HDDT 2 / 2





Split criterion: Hellinger Distance (HD)

- Gini index and information gain are skew sensitive, unlike HD
- For continuous distributions P, Q of normalized feature frequency values, HD is defined as

$$d_H(P,Q) = \sqrt{\int_\Omega(\sqrt{p}-\sqrt{q})^2 d\lambda} = \sqrt{2(1-\int_\Omega \sqrt{pq}\, d\lambda)}$$

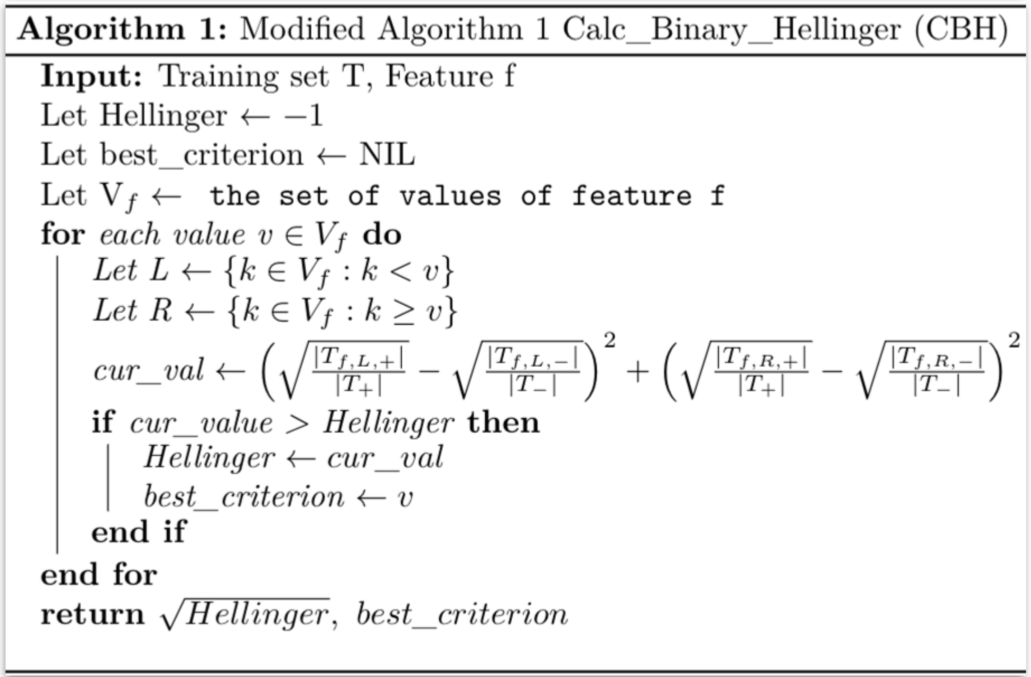
- Discrete case:

$$\boldsymbol{HD = d_H(X_+, X_-) = \sqrt{\sum_{j=1}^{p}\left(\sqrt{\frac{|X_{+j}|}{|X_+|}} - \sqrt{\frac{|X_{-j}|}{|X_-|}}\right)^2}}$$

- HD takes values in [0, √2]: **"The aim is to split tree nodes on those features with minimal *affinity* i.e. maximal HD."**

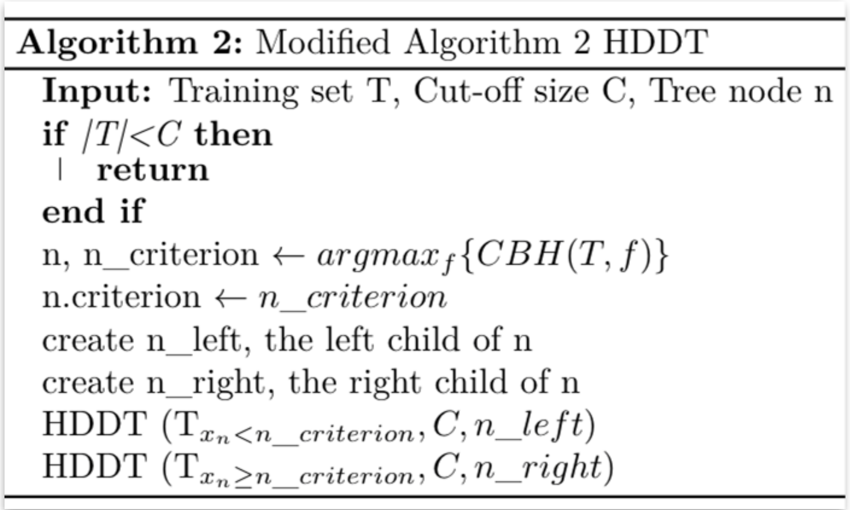
Constructing the HDDT

- Labeled dataset (*n* training samples): $D_n = \{ (X_i , Y_i), i = 1, 2, \dots , n \}, Y \in \{0, 1\}$
- Normalized *p* input features: $X \in C^p = [0, 1]^p$; e.g. $X = (0.345, 0.582, 0.291, \dots)$



Algorithm 1: Modified Algorithm 1 Calc_Binary_Hellinger (CBH)

```
Input: Training set T, Feature f
Let Hellinger ← −1
Let best_criterion ← NIL
Let $V_f$ ← the set of values of feature f
for each value $v \in V_f$ do
    Let $L \leftarrow \{k \in V_f : k < v\}$
    Let $R \leftarrow \{k \in V_f : k \geq v\}$
    $cur\_val \leftarrow \left(\sqrt{\frac{|T_{f,L,+}|}{|T_+|}} - \sqrt{\frac{|T_{f,L,-}|}{|T_-|}}\right)^2 + \left(\sqrt{\frac{|T_{f,R,+}|}{|T_+|}} - \sqrt{\frac{|T_{f,R,-}|}{|T_-|}}\right)^2$
    if $cur\_value > Hellinger$ then
        $Hellinger \leftarrow cur\_val$
        $best\_criterion \leftarrow v$
    end if
end for
return $\sqrt{Hellinger}$, $best\_criterion$
```



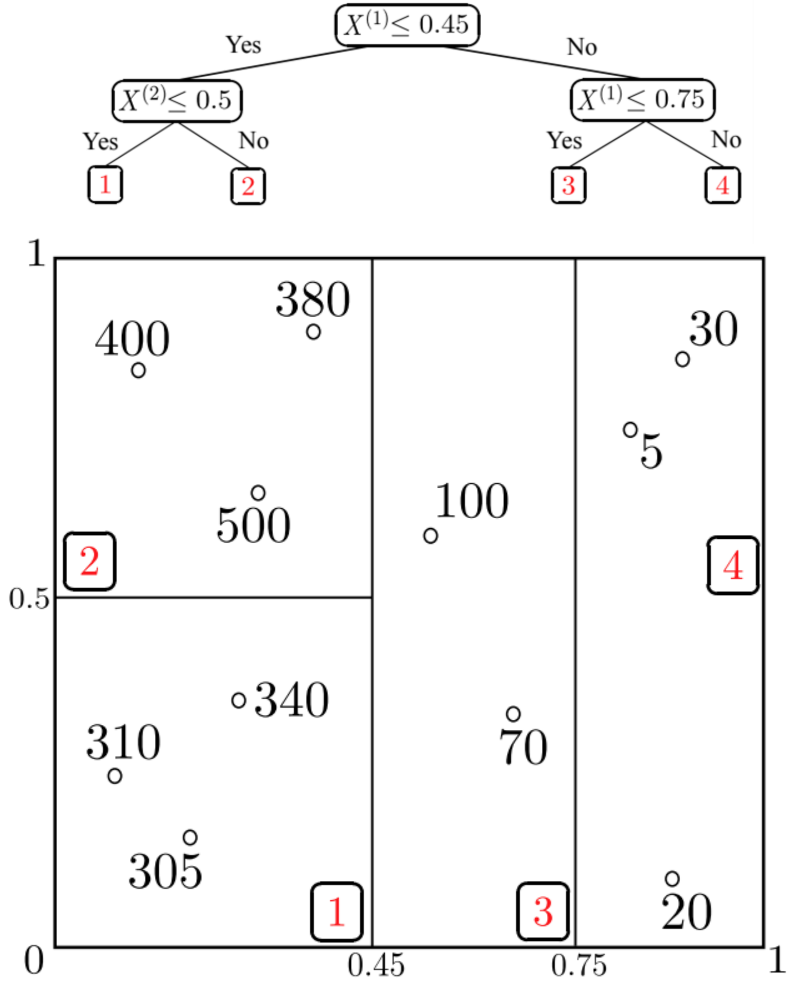
Algorithm 2: Modified Algorithm 2 HDDT

```
Input: Training set T, Cut-off size C, Tree node n
if $|T| < C$ then
    return
end if
n, n_criterion ← $argmax_f\{CBH(T, f)\}$
n.criterion ← $n\_criterion$
create n_left, the left child of n
create n_right, the right child of n
HDDT ($T_{x_n < n\_criterion}, C, n\_left$)
HDDT ($T_{x_n \geq n\_criterion}, C, n\_right$)
```

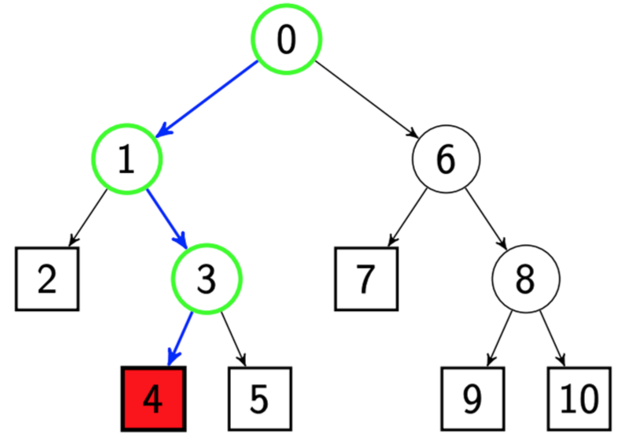
Traversing the HDDT

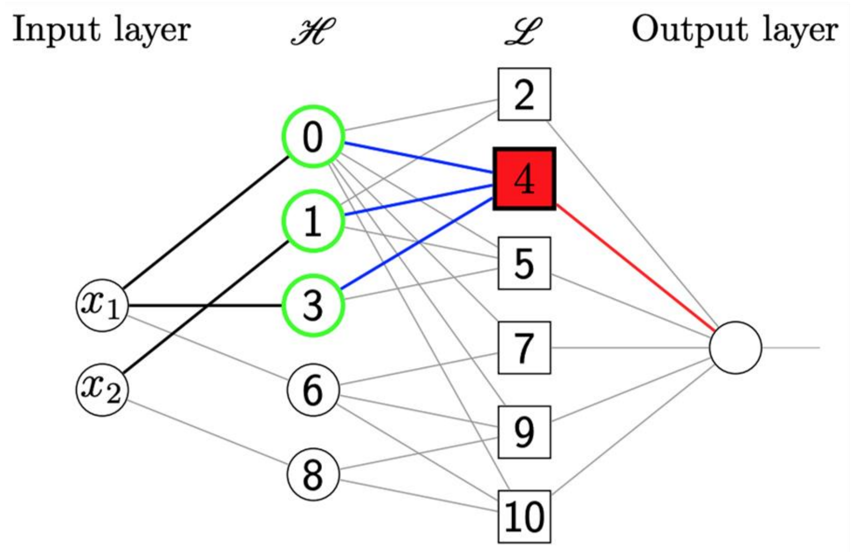
- Input ***x*** passed to the tree root, then iteratively towards the leaf node representing area ***S***
- Tree estimate $t_n(x)$ (0/0 = 0 by convention):

$$t_n(x) = \frac{1}{N_n(S)}\left(\sum_{\{x_i \in S\}} Y_i\right)$$



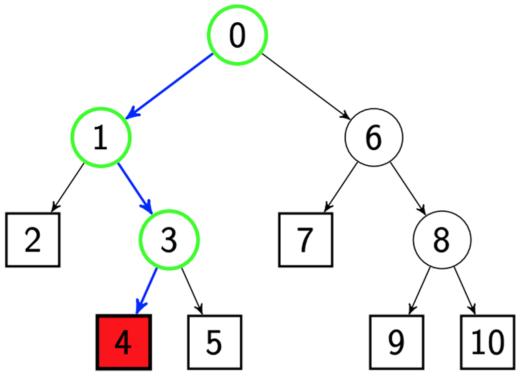
From HDDT to Hellinger Net

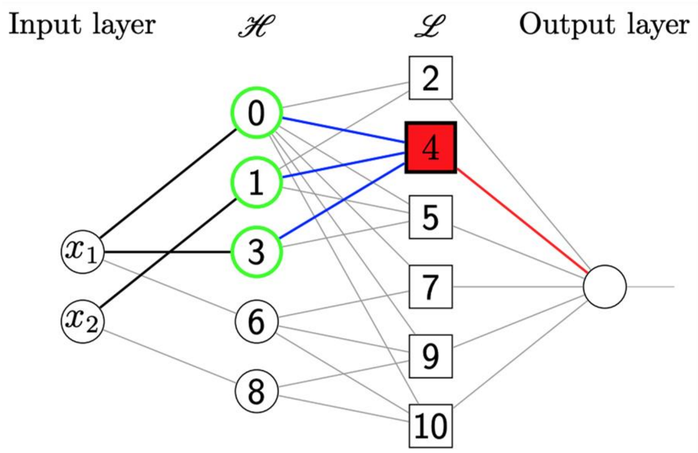




From HDDT to Hellinger Net: Input & First Hidden Layer

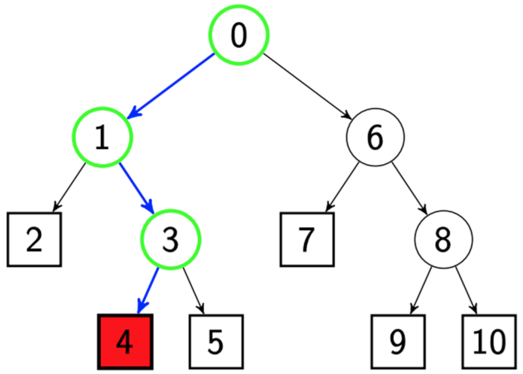
- The input layer supplies the features (labeled $\boldsymbol{x}_i$ or $\boldsymbol{x}^{(i)}$) to HL1, which corresponds to k–1 perceptrons (since our HDDT is a full binary tree, it has *k–1* internal nodes, and *k* leaves)
- The first hidden layer (HL1) consists of hyperplanes $H_{k'} = \{x \in C^p : h_{k'}(x) \geq 0\}$ with $h_{k'}(x) = x^{(i_{k'})} - \alpha_{i_{k'}}$ and we're trying to find the side on which $\boldsymbol{x}$ falls
- The threshold activation function is applied $\tau(u) = 2 \times I_{u \geq 0} - 1$

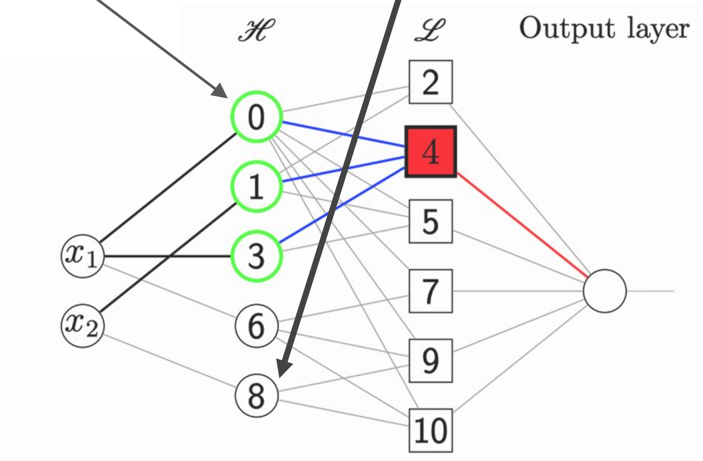




First Hidden Layer output

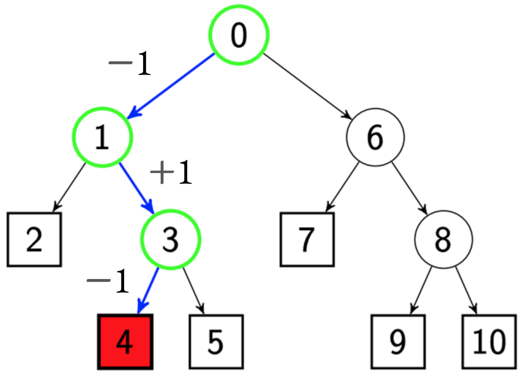
- By applying the threshold activation function $\tau(h_{k'}(x)) = \tau(x^{(i_{k'})} - \alpha_{i_{k'}})$ each split and relative position of ***x*** in the HDDT is encoded in each neuron, so HL1 outputs a **vector of ±1 bits**, $(\tau(h_1(\mathbf{x})), \tau(h_2(\mathbf{x})), \ldots, \tau(h_{k-1}(\mathbf{x})))$, +1 if ***x*** falls on the right side (h(***x***) ≥ 0), and −1 otherwise.

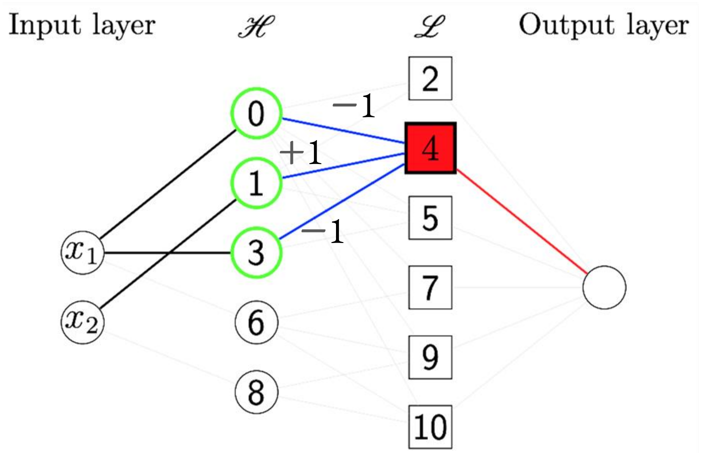




Second Hidden Layer: partition ANDing 1/2

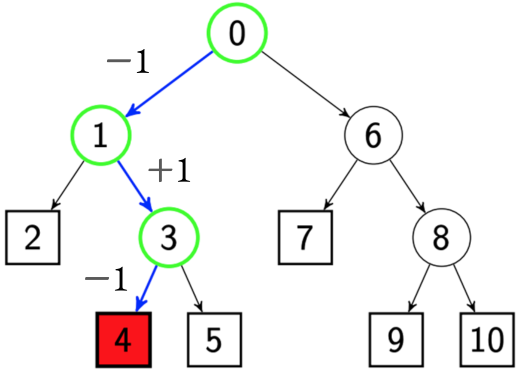
- The second hidden layer (HL2) comprises k leaves, and each connection from $\boldsymbol{k'}$ to $\boldsymbol{k''}$ has weight $b = +1$ if in order for $\boldsymbol{x}$ to get to $\boldsymbol{k''}$, it follows the right branch of $\boldsymbol{k'}$ in HDDT, or $b = -1$ if it follows the left branch.

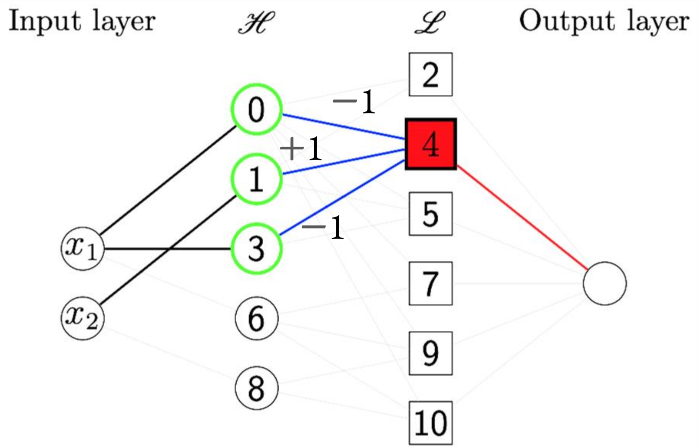




Second Hidden Layer: partition AND-ing 2/2

- Get $(u_1(x), \dots, u_{k-1}(x))$ from HL1 (a vector of ±1 bits)
- $$u = \begin{cases} \sum_{k' \to k''} \boxed{b_{k',k''} u_{k'}(x)} - \boxed{l(k'')} + \frac{1}{2} = \frac{1}{2} & \text{if } x \in L_{k''} \\ \sum_{k' \to k''} \boxed{b_{k',k''} u_{k'}(x)} - \boxed{l(k'')} + \frac{1}{2} \le -\frac{1}{2} & \text{if } x \notin L_{k''} \end{cases}$$
- Apply the threshold activation function *τ(u)* ⇒ "one-hot" vector with −1s instead of 0s





Output Layer: "one-hot" vector OR-ing

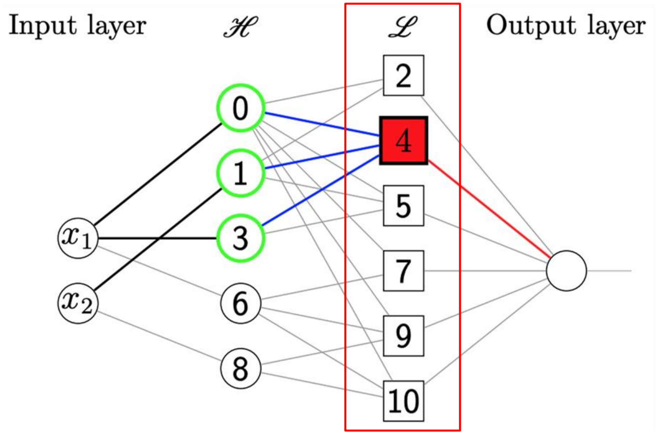
$$t_n(x) = \sum_{k''=1}^{k} \boxed{w_{k''}}\boxed{v_{k''}(x)} + \boxed{b_{\text{out}}} \quad \text{with} \quad \boxed{w_{k''} = \frac{\bar{Y}_{k''}}{2}} \boxed{b_{out} = \frac{1}{2}\sum_{k''=1}^{k} \bar{Y}_{k''}}$$

$$= \sum_{k''=1}^{k} v_{k''}(x)\bar{Y}_{k''}\frac{1}{2} + \frac{1}{2}\sum_{k''=1}^{k} \bar{Y}_{k''}$$

$$= \frac{1}{2}\sum_{k''=1}^{k} \bar{Y}_{k''}\left(v_{k''}(x) + 1\right)$$

$$= \bar{Y}_{k''}$$

$$g_n(x) = \begin{cases} 0, & \text{if} \quad t_n(x) \leq \frac{1}{2} \\ 1, & \text{otherwise} \end{cases}$$

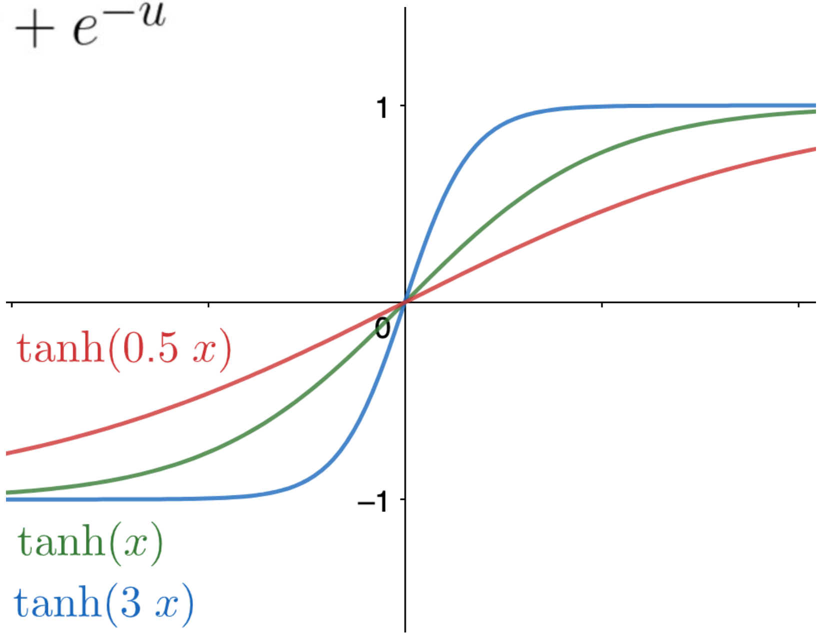


Finishing touches

- The Hellinger net uses a **sigmoid** activation function instead of a threshold, relay-type one

$$\tau(u) = \sigma(u) = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$

- HL1 uses $\sigma(\beta_1 u)$, while HL2 uses $\sigma(\beta_2 u)$
- The Hellinger net is trained using **stochastic gradient descent backpropagation** algorithm
- The network requires fitting $O(p \times k + k^2)$ parameters, and if the HDDT is roughly balanced, then $O(k\log(k))$.



Hellinger Net Algorithm

- Build a HDDT with $(k_n - 1)$ split nodes and k_n leaf nodes. HDDT is mapped into a two hidden layered MLP model having $(k_n - 1)$ and k_n hidden neurons in first hidden layer $(HL1)$ and second hidden layer $(HL2)$, respectively.
- The first hidden layer is called the partitioning layer which partitions the input feature spaces into different regions. It corresponds to the internal nodes of the DT. In $HL1$, the neurons compute all the tree split decisions and indicate the split directions for the inputs.
- Further, $HL1$ passes the information to $HL2$. The neurons in the second hidden layer represent the terminal nodes of the DT.
- The final layer is the output class label of the tree. Train the tree structured neural network using gradient descent backpropagation algorithm.
- Hellinger Net uses sigmoidal activation function instead of the relay-type activation function $\tau(u)$ with a hyperbolic tangent activation function $\sigma(u) = \tanh(u)$ which has a chosen range from -1 to 1.
- More precisely, the model uses $\sigma_1(u) = \sigma(\beta_1 u)$ at every neuron of the first hidden layer for better generalization, where β_1 is a positive hyper-parameter that determines the contrast of the hyperbolic tangent activation function.

Merits

- **Merits**:

  1. The additional training using backpropagation potentially improves the predictions of the HDDT and can deny tree pruning steps vis-a-vis the risk of overfitting.;

  2. Hellinger Nets give weight to nodes according to their significance as determined by the gradient backpropagation algorithm.;

  3. In Hellinger Nets, the neural network follows the built-in hierarchy of the originating tree since connections do not exist between all pairs of neurons in any two adjacent layers.;

  4. Since the number of neurons in the hidden layers are fixed, thus the training time is less.

- **Theoretical developments**:

  1. Theoretical Consistency?

  2. Rate of Convergence?

On Theoretical Consistency

Theorem (Chakraborty et al., 2020, IEEE Transactions on Reliability)

Assume X is uniformly distributed in $[0,1]^p$ and $Y = \{0,1\}$. As $n \to \infty$ and for any $k_n, \beta_1, \beta_2 \to \infty$ if the following conditions are satisfied:

$$(A1) \quad \frac{k_n^4 log(\beta_2 k_n^4)}{n} \to 0,$$

$$(A2) \quad \textit{there exists} \quad \delta > 0 \quad \textit{such that} \quad \frac{k_n^2}{n^{1-\delta}} \to 0,$$

$$(A3) \quad \frac{k_n^2}{e^{2\beta_2}} \to 0, \quad \textit{and}$$

$$(A4) \quad \frac{k_n^3 \beta_2}{\beta_1} \to 0,$$

then Hellinger Nets classifier is consistent.

The above Theorem states that with certain restrictions imposed on the number k_n of terminal nodes and the parameters β_1, β_2 being properly regulated as functions of n, the empirical L_1 risk-minimization provides local consistency of the Hellinger Nets classifier.

Rate of Convergence

Theorem (Chakraborty et al., 2020, IEEE Transactions on Reliability)

Assume that X is uniformly distributed in $[0,1]^p$ and $Y = \{0,1\}$ and a function $m : C^p \to \{0,1\}$ is a Lipschitz $(\delta; C)$-smooth for any $\delta \in [0,1]$. Let m_n be the estimate that minimizes empirical L_1-risk and the network activation function σ_i satisfies Lipschitz property. Then for any $n \geq max\{\beta_2, 2^{p+1}L\}$, we have

$$E \int_{[0,1]^p} \left| m_n(X) - m(X) \right| \mu(dx) = O\left(\frac{log(n)^6}{n} \right)^{\frac{2}{2+p}}$$

- The proof of the Theorem is using Complexity Regularization Principles.
- The rate of convergence doesn't depend on the data dimension and hence the model will be able to circumvent the so-called problem of "curse of dimensionality".
- In practice, the larger the value of k_n, β_1, and β_2, the better the model performance is.

Applications: Data Sets

Data Sets: The proposed model is evaluated using five publicly available UCI data sets.

Table: Characteristics of the UCI data sets used in experimental evaluation

Data set	Classes	Objects (n)	Number of feature (p)	Number of $(+)$ve instances	Number of $(-)$ve instances	CV
breast cancer	2	286	9	201	85	0.41
german credit card	2	1000	20	700	300	0.40
indian business school	2	480	17	400	80	0.56
page blocks	2	5473	10	4913	560	0.80
pima diabetes	2	768	8	500	268	0.30

Performance Evaluation

The performance evaluation measure used in our experimental analysis is based on the confusion matrix in Table. Area under the receiver operating characteristic curve (AUC) is a popular metric for evaluating performances of imbalanced data sets and higher the value of AUC, the better the classifier is. AUC $= \frac{\text{Sensitivity}+\text{Specificity}}{2}$; where, Sensitivity $= \frac{TP}{TP+FN}$; Specificity $= \frac{TN}{FP+TN}$.

Table: AUC results (and their standard deviation) of classification algorithms over original imbalanced test data sets

Classifiers	breast cancer	German credit card	Indian business school	page blocks	pima diabetes
CT	0.603 (0.04)	0.665 (0.03)	0.810 (0.04)	0.950 (0.00)	0.724 (0.02)
RF	0.690 (0.06)	0.725 (0.03)	0.850 (0.04)	0.964 (0.00)	0.747 (0.04)
k-NN	0.651 (0.03)	0.727 (0.01)	0.750 (0.03)	0.902 (0.02)	0.730 (0.05)
RBFN	0.652 (0.06)	0.723 (0.04)	0.884 (0.05)	0.935 (0.01)	0.725 (0.04)
HDDT	0.625 (0.04)	0.738 (0.04)	0.933 (0.02)	0.974 (0.00)	0.760 (0.02)
HDRF	0.636 (0.04)	0.742 (0.03)	0.939 (0.02)	**0.988** (0.00)	0.760 (0.03)
ANN (with 1HL)	0.585 (0.03)	0.700 (0.03)	0.768 (0.05)	0.918 (0.02)	0.649 (0.03)
ANN (with 2HL)	0.621 (0.02)	0.715 (0.02)	0.820 (0.04)	0.925 (0.01)	0.710 (0.03)
Hellinger net	**0.730** (0.05)	**0.802** (0.03)	**0.968** (0.01)	0.980 (0.02)	**0.809** (0.03)

Simulation Study

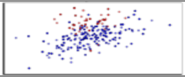
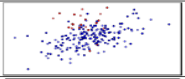
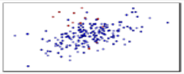
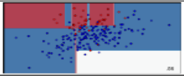
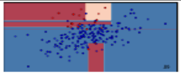
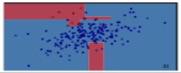
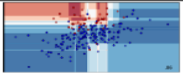
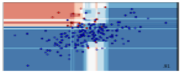
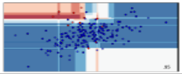
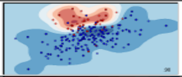
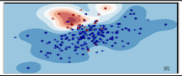
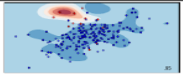
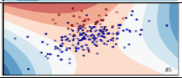
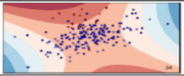
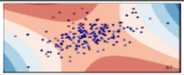
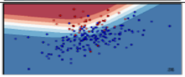
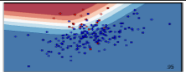
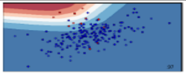
Simulated Data Sets: Three toy data sets (binary) are generated with weights = [0.2, 0.8], [0.1, 0.9] and [0.05, 0.95], i.e., data sets with imbalance rates of 20%, 10% and 5%, respectively. We added Gaussian noise to the data with the standard deviation equals to 0.5. This test problem is suitable for algorithms that can learn data imbalance problems in complex nonlinear manifolds.

Table: AUC results of different imbalanced classifiers on three synthetic data sets.

Imbalanced Classifiers	Simulated Data with IR = 20%	Simulated Data with IR = 10%	Simulated Data with IR = 5%
HDDT	0.80	0.85	0.91
HDRF	0.82	0.88	0.91
VCB-SVM	**0.87**	0.89	0.93
ISDA	0.84	0.91	0.90
Hellinger net	0.86	**0.92**	**0.95**

Simulation Study

A comparison of several imbalanced classifiers on synthetic data sets. The plots show training points in solid colors and testing points semi-transparent. The lower right in each plots shows the classification accuracy on the test set.

Synthetic data	Imbalance = 20% rate	Imbalance = 10% rate	Imbalance = 5% rate
Input Data			
HDDT			
HDRF			
VCB-SVM			
ISDA			
Hellinger net			

Conclusions

- Learning from an imbalanced data set presents a tricky problem in which traditional learning models perform poorly.
- Simply allocating half of the training examples to the minority class does not provide the optimal solution in most of the real-life problems.
- If one would like to work with the original data without taking recourse to sampling, our proposed hybrid methodology will be quite handy.
- We proposed 'Hellinger Nets', a hybrid learner, that first construct a tree and then simulate it using neural networks.
- We have proved the consistency of Hellinger Net model.

Future Work

- The scope of future research of this work will be to improve the proposed Hellinger net model for imbalanced classification problem with concept shift in the data sets.
- Another scope of future research of the thesis will be to build Hybrid Models for Adversarial Machine Learning Problems.
- Use of Wasserstein Distance is of much use in the Machine Learning community for the last few decades. Some modification to the Wasserstein distance can be done and incorporated in the DT, RF, and Hellinger net model. This may improve the existing HDDT, HDRF and Hellinger Net models for imbalanced pattern classification.

Thank You