The Deep Forest and its Modifications

Lev V. Utkin

pml.spbstu.ru

Munich, November 4, 2017

In memory of Kurt Weichselberger

Peter the Great Saint-Petersburg Polytechnic University

Agenda

1. Decision Trees
2. Random Forest
3. Stacking
4. **Deep Forest** (gcForest)
5. Improved deep forest (IDF)
6. Siamese deep forest (SDF) and discriminative DF (DisDF)
7. Transfer learning (TLDF)

Decision Trees

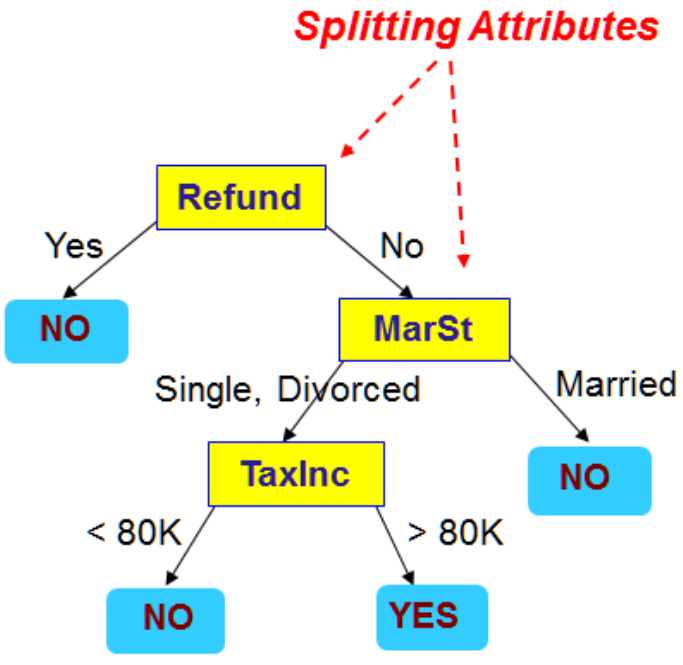
Decision Trees (definition)

- Decision tree is a **classifier** in the form of a tree structure, where each node is either:
  - Decision node - specifies some test to be carried out on a single attribute-value, with one branch and subtree for each possible outcome of the test
  - Leaf node - indicates the value of the target attribute (class) of examples
- Decision trees attempt to **classify** a pattern through a sequence of questions.
- CART (Classification And Regression Trees)

Decision Trees (example)

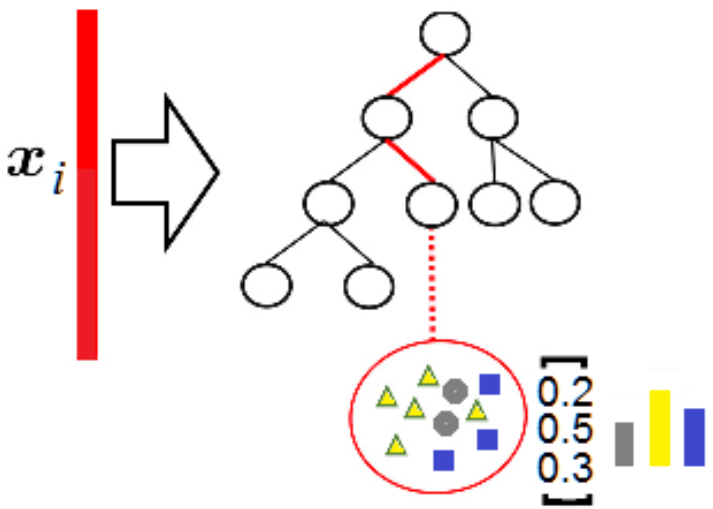
The problem of predicting whether a loan will reply her loan obligations or become delinquent, subsequenlty defaulting on her loan.

Tid	Refund	Marital Status	Taxable Income	Cheat (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Very important

For every example, we can compute the probabilities of classes **by counting the percentage of different classes of examples at the leaf node where the concerned instance falls into!**



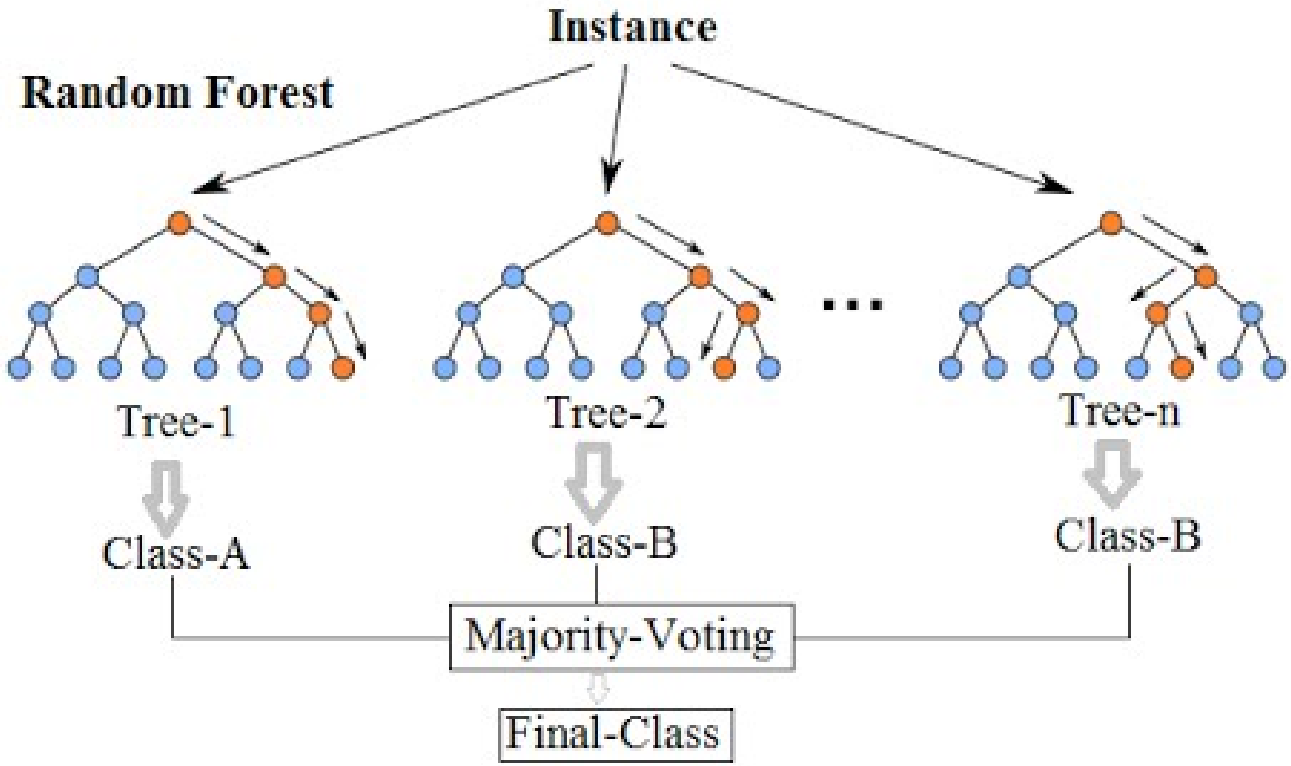
Random Forest

Random Forest

Random Forest

- Combination of the bagging (random selection of examples) and random selection of features
- Random Forest grows many classification trees.
- Each tree gives a classification, i.e., it "votes" for that class.
- The forest chooses the classification having the most votes (over all the trees in the forest).

Random Forest



Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data.
2. If there are M input features, then $m \ll M$ features are selected at random out of the M and the best split on these m is used to split the node.
3. Each tree is grown to the largest extent possible. There is no pruning.

Overfitting problem

Random forest does not overfit (Leo Breiman and Adele Cutler)

https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Stacking

Stacking (Wolpert, 1992)

The basic idea is the following algorithm:

1. to train the **first-level learners** using the original training data set
2. generate a new data set for training the **second-level learner (meta-learner)**
3. the outputs of the first-level learners are regarded as input features for the second-level learner while the original labels are still regarded as labels of the new training data.

A general algorithm of stacking

Input: Training data $S = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$

1. Learn T first-level classifiers h_t based on S
2. Construct new data set S^* from S:
   $S^* = \{(\mathbf{x}_1^*, y_i), ..., (\mathbf{x}_n^*, y_i)\}$, where $\mathbf{x}_i^* = \{h_1(\mathbf{x}_i), ..., h_T(\mathbf{x}_i)\}$
3. Learn a second-level classifier by using the new data set S^*

Output: An ensemble classifier $H(\mathbf{x}) = h^*(h_1(\mathbf{x}), ..., h_T(\mathbf{x}))$

Stacking example (1)

	f_1	f_2	f_3	f_4	y	y_1	y_2	y_3
	cough	t, C	age	snuffle	diagnosis	1	2	3
x_1	slight	36.6	42	yes	cold	c	c	f
x_2	bad	39.1	25	no	flu	c	f	c
x_3	bad	37.8	30	yes	cold	c	c	c
x_4	slight	38.2	47	yes	cold	f	f	c
...	...	...	...	...	...	...	...	...
x_{100}	slight	38.4	32	no	flu	f	f	c

Stacking example (2)

cold $\longrightarrow$ 1; flu $\longrightarrow$ -1

	f_1	f_2	f_3	f_4	$f_5 = \frac{y_1+y_2+y_3}{3}$	y
	cough	t, C	age	snuffle	diagnosis1	diagn.
x_1	slight	36.6	42	yes	1/3	cold
x_2	bad	39.1	25	no	1/3	flu
x_3	bad	37.8	30	yes	1	cold
x_4	slight	38.2	47	yes	-1/3	cold
...	...	...	...	...	...	...
x_{100}	slight	38.4	32	no	-1/3	flu

Deep Forest (gcForest)

Deep Forest

Deep Forest is a musical group originally consisting of two French musicians

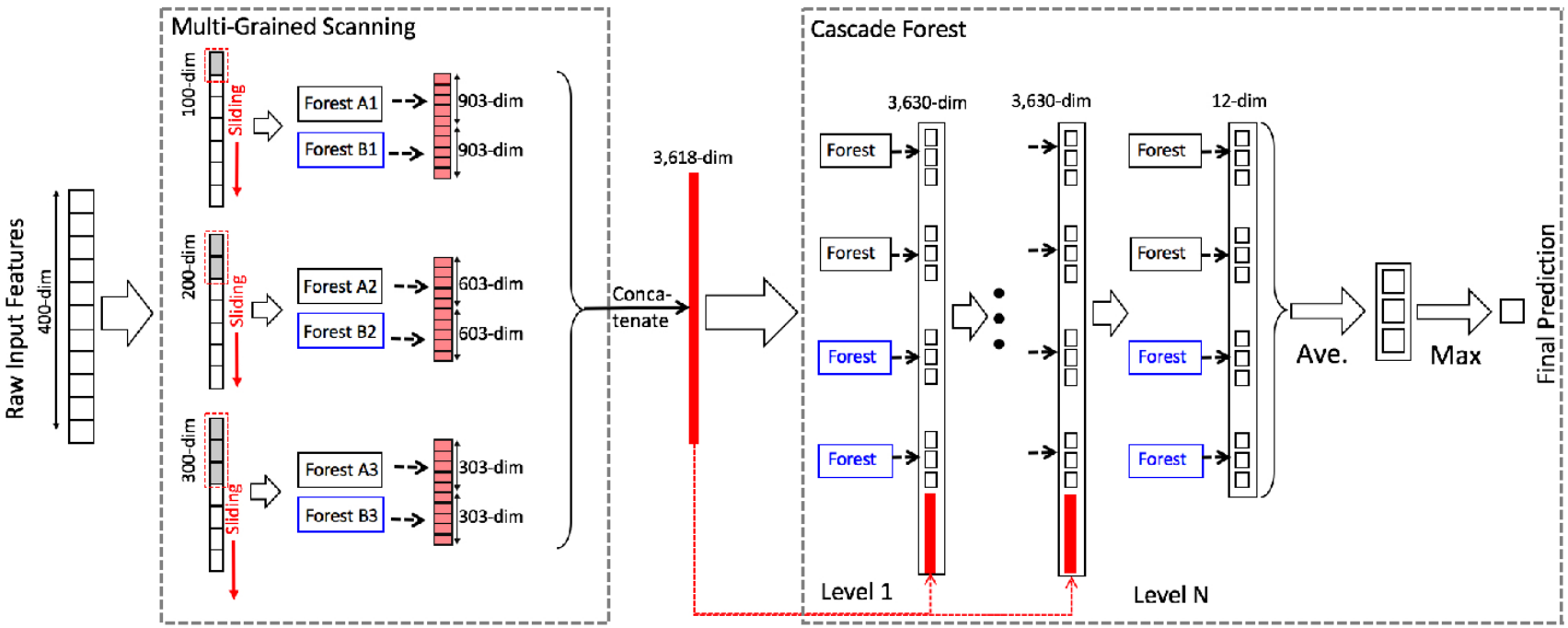
Deep Forest

Zhi-Hua Zhou and Ji Feng. Deep Forest: Towards An Alternative to Deep Neural Networks // arXiv:1702.08835v1 [cs.LG] 28 Feb 2017.

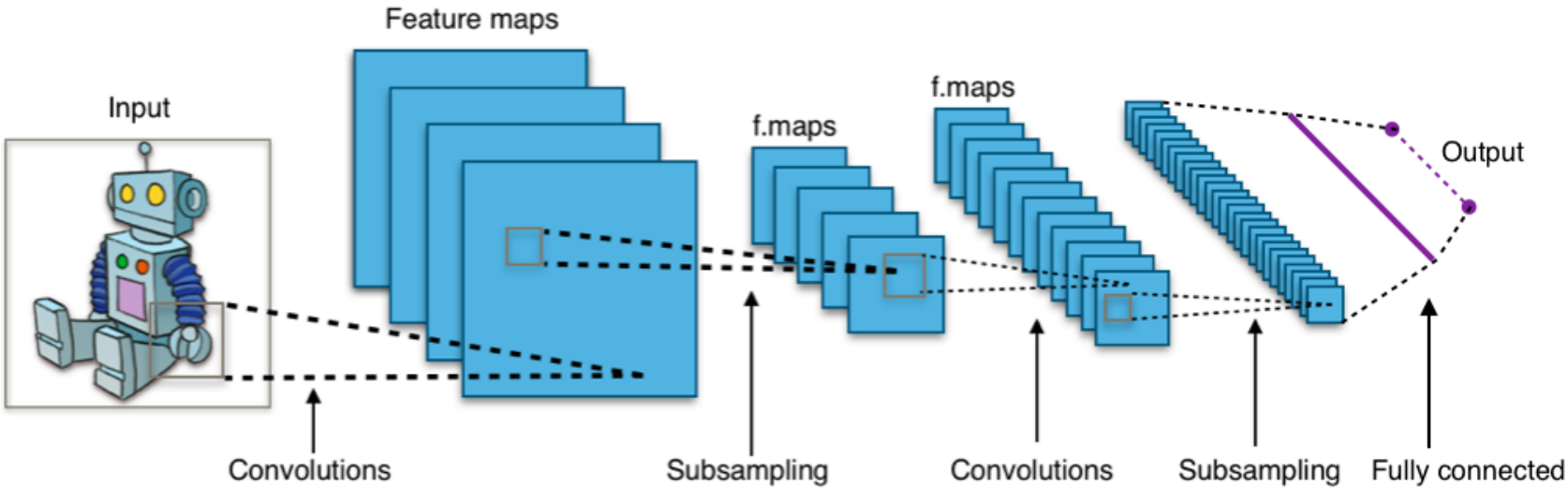
Phrases from Abstract:

- ... we propose gcForest, a decision tree ensemble approach with performance highly competitive to deep neural networks.
- ... in contrast to DNN, gcForest is much easier to train.
- ... in contrast to DNN which require large-scale training data, gcForest can work well even when there are only small-scale training data.

Overall Structure

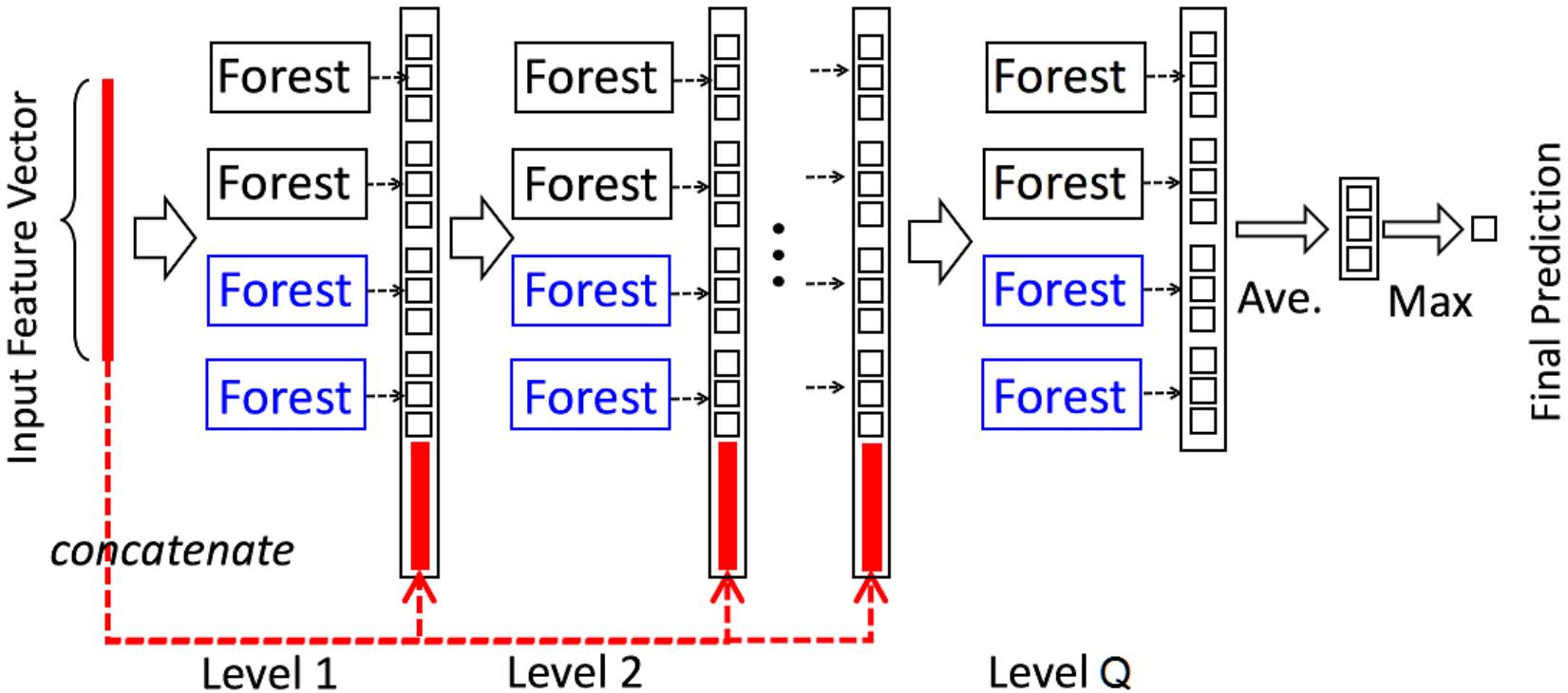


Deep convolution neural network



https://en.wikipedia.org/wiki/Convolutional_neural_network

Cascade Forest (architecture)

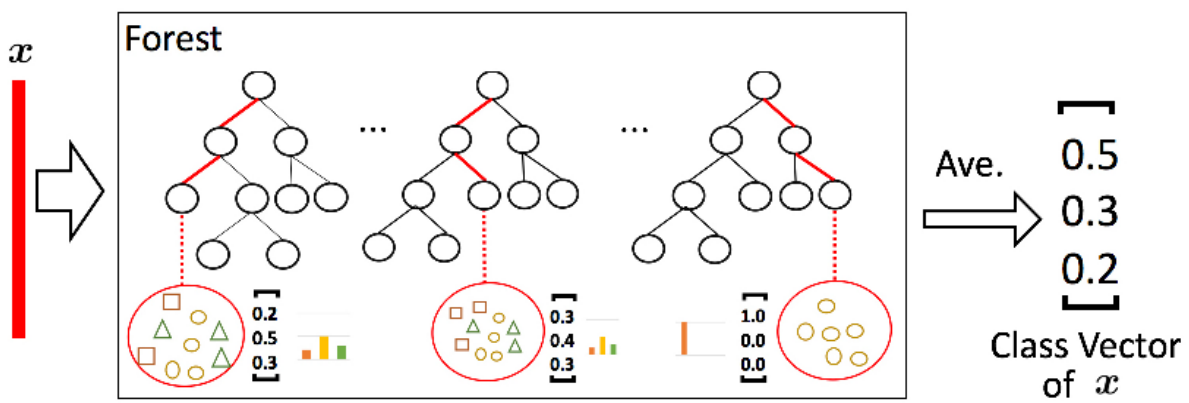


Cascade Forest

- Representation learning in deep neural networks mostly relies on the layer-by-layer processing of raw features
- gcForest employs a cascade structure, where each level of cascade receives feature information processed by its preceding level, and outputs its processing result to the next level
- Each level is an ensemble of random forests, i.e., an **ensemble of ensembles**
- To encourage the diversity, two **complete-random tree forests** and two **random forests** are used

Class vector generation

- Given an instance, each forest produces an estimate of class distribution:
  - **by counting the percentage of different classes of examples at the leaf node where the concerned instance falls into,**
  - then averaging across all trees in the same forest
- The class distribution forms a **class vector**, which is then **concatenated with the original vector** to be input to the next level of cascade.



Example of concatenation (1)

	f_1	f_2	f_3	f_4	y
	cough	t, C	age	snuffle	diagnosis
x_1	slight	36.6	42	yes	cold
x_2	bad	39.1	25	no	flu
x_3	bad	37.8	30	yes	cold
x_4	slight	38.2	47	yes	cold
...	...	...	...	...	...
x_{100}	slight	38.4	32	no	flu

Example of concatenation (2)

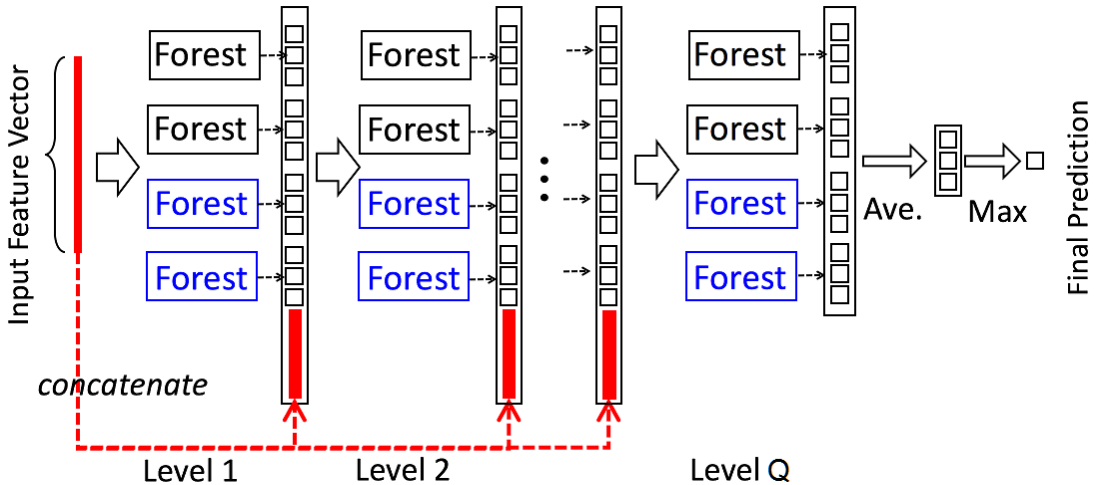
f_1	f_2	f_3	f_4	y	Tree 1	Tree 2	Forest 1	
cough	t, C	age	snuffle	diag.	pr. (c,f)	pr. (c,f)	Ave. 1	
slight	36.6	42	yes	cold	(0.9;0.1)	(0.7;0.3)	0.8	0.2
bad	39.1	25	no	flu	(0.3;0.7)	(0;1)	0.15	0.85
bad	37.8	30	yes	cold	(0.6;0.4)	(0.9;0.1)	0.75	0.25
slight	38.2	47	yes	cold	(0.8;0.2)	(0.8;0.2)	0.8	0.2
...	...	...	...	...	...	...	...	
slight	38.4	32	no	flu	(0.1;0.9)	(0.5;0.5)	0.3	0.7

Example of concatenation (3)

				Forest 1		Forest 2		Forest 3	
cough	t, C	age	snuffle	Ave. 1		Ave. 2		Ave. 3	
f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
slight	36.6	42	yes	0.8	0.2	1	0	0.75	0.25
bad	39.1	25	no	0.15	0.85	0.21	0.79	0.18	0.82
bad	37.8	30	yes	0.75	0.25	0.85	0.15	0.5	0.5
slight	38.2	47	yes	0.8	0.2	0.9	0.1	0.8	0.2
...	...	...	...	...					
slight	38.4	32	no	0.3	0.7	0.14	0.86	0.22	0.78

Cascade Forest

For 3 classes and 4 forests which produce a three-dimensional class vector - the next level of cascade will receive $12 = 3 \times 4$ augmented features.



Number of features generated by a level = number of forests $\times$ number of classes

Improved Deep Forest (IDF)

A shortcoming of gcForest

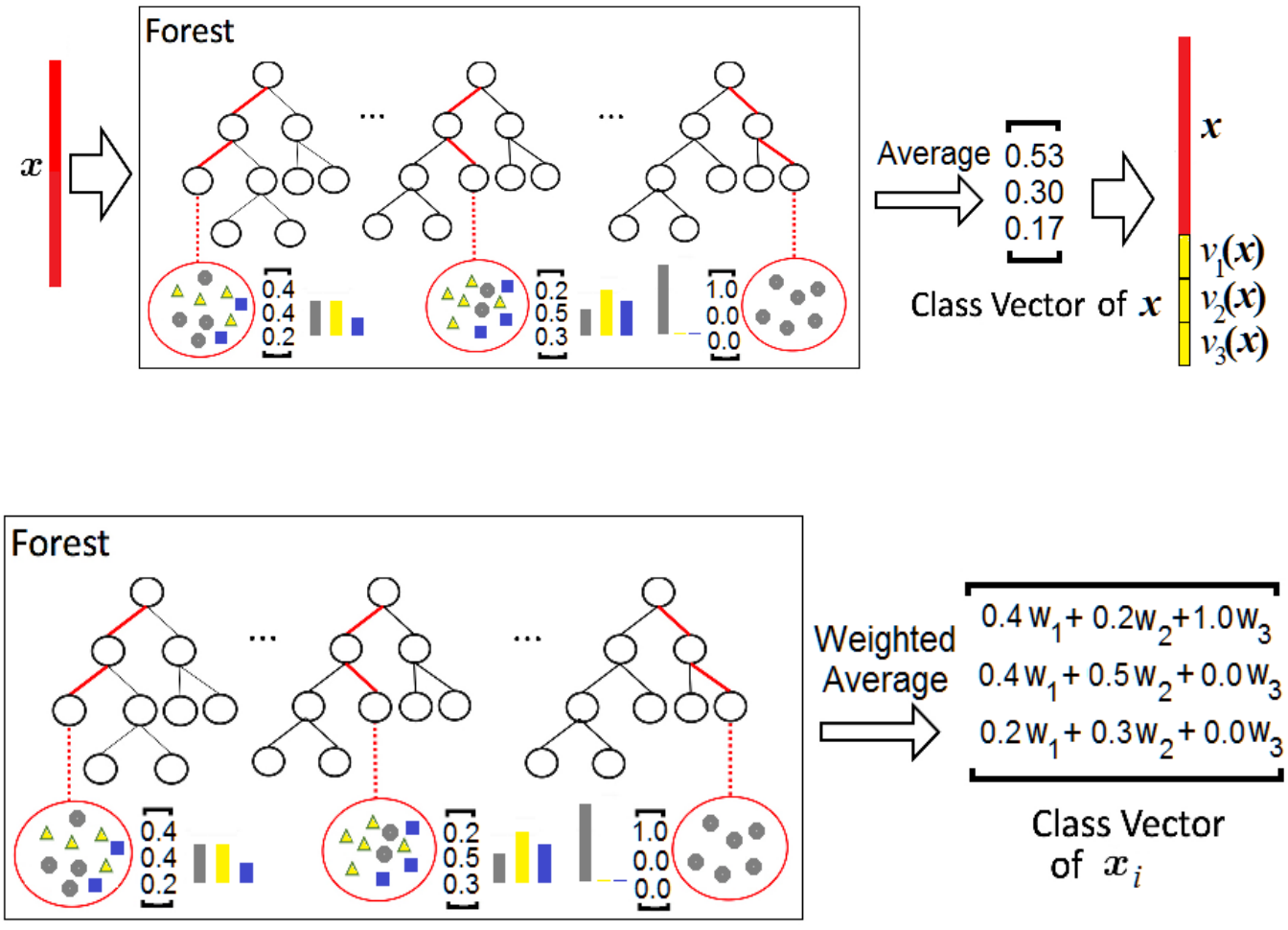
In contrast to neural networks gcForest cannot be controlled, i.e., we cannot define a loss function for solving problems different from the standard classification, in particular, distance metric learning, transfer learning and domain adaptation, etc.

How to improve the deep forest?

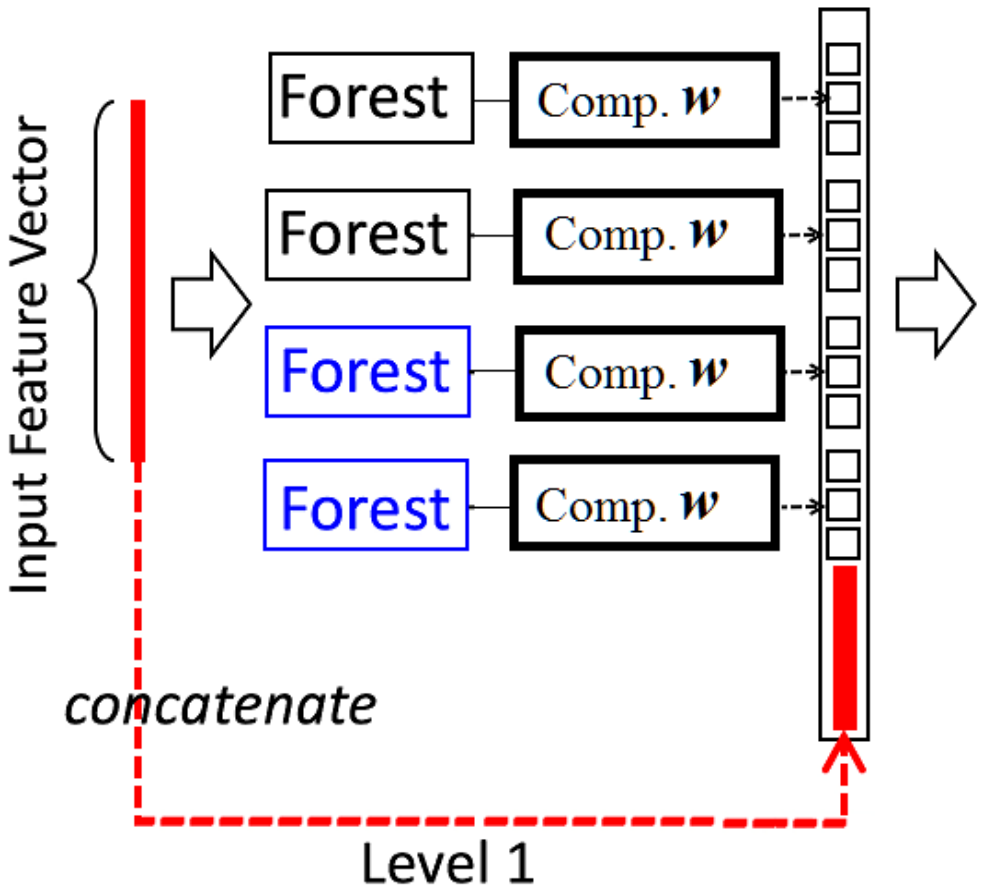
Main ideas:

1. We can change the class vectors (augmented features) at every level in accordance with a goal, for example, to concentrate the semantically similar objects
2. We can do it by introducing weights of trees and computing a weighted average instead of the ordinary average
3. The weights are not assigned by someone. They are trained jointly with trees on the basis of training data
4. So, we can control the deep forest through control of the class vectors

Class vectors as weighted sums



Additional components to the deep forest



Weights

- New augmented features

$$v_{i,c} = \mathbf{p}_{i,c}\mathbf{w} = \sum_{t=1}^{T} p_{i,c}^{(t)} w_t$$

- Condition for weights

$$\sum_{t=1}^{T} w_t = \mathbf{w} \cdot \mathbf{1}^{\mathrm{T}} = 1, \;\; w_t \geq 0, \;\; t = 1, ..., T$$

- Weights do not depend on classes, on training examples $\mathbf{x}_i$
- We cannot construct ideal trees, but we can supplement the trees by a procedure that could try to approximate the class prob. distribution of the RF by a goal class vector

Goal class vector (simplest case)

- An ideal tree is when the class probability distribution for $\mathbf{x}_i$ by $y_i = c$ at a leaf node is

$$\mathbf{o}_i = (0, ..., 0, 1_c, 0, ..., 0)$$

- The loss function is the distance between the class vector $\mathbf{v}_i$ and the vector $\mathbf{o}_i$:

$$J_{q,k}(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^{N} d(\mathbf{v}_i, \mathbf{o}_i) + \lambda R(\mathbf{w}).$$

subject to $\mathbf{w} \cdot \mathbf{1}^{\mathrm{T}} = 1, \ w_t \geq 0.$

- The Euclidean distance

$$J_{q,k}(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^{N} \sum_{c=1}^{C} \left(\mathbf{p}_{i,c}\mathbf{w} - \mathbf{1}(y_i = c)\right)^2 + \lambda \left\|\mathbf{w}\right\|^2$$

Frank-Wolf algorithm

1. Initialize $w_0 \in \Delta$ is the unit simplex with T vertices
2. **FOR** $j = 0$, $j \leq S - 1$
3. Compute $\nabla_{w_r} J_{q,k}(w)$ for every $r = 1, ..., T$
4. Compute $t_0 = \arg\min_{t=1,...,T} \nabla_{w_t} J_{q,h}(w)$
5. $\mathbf{g}_j \leftarrow (0, ..., 0, 1_{t_0}, 0, ..., 0)$
6. Compute $\gamma_j \leftarrow 2/(j+2)$
7. Update $w_{j+1} \leftarrow w_j + \gamma_j (\mathbf{g}_j - \mathbf{w}_j)$
8. **END of the Loop**

MNIST numerical experiments

Table: The IDF accuracy for the MNIST data set by different N and T in every forest

T	100		700		1000	
N	gcF	IDF	gcF	IDF	gcF	IDF
300	0.700	0.740	0.750	0.760	0.700	0.750
500	0.802	0.826	0.784	0.808	0.790	0.800
1000	0.856	0.862	0.850	0.850	0.850	0.850
2000	0.884	0.886	0.895	0.899	0.876	0.880

Siamese Deep Forest (SDF)

Distance metric learning problem statement

- Training set $S = \{(\mathbf{x}_i, \mathbf{x}_j, y_{ij}),\ (i,j) \in K\}$ from N pairs $\mathbf{x}_i \in \mathbb{R}^m$ and $\mathbf{x}_j \in \mathbb{R}^m$, $y_{ij} \in \{0, 1\}$
- Semantically similar objects

$$\mathcal{S} = \{(\mathbf{x}_i, \mathbf{x}_j) : y_{ij} = 0\}.$$

- Semantically dissimilar objects

$$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{x}_j) : y_{ij} = 1\}.$$

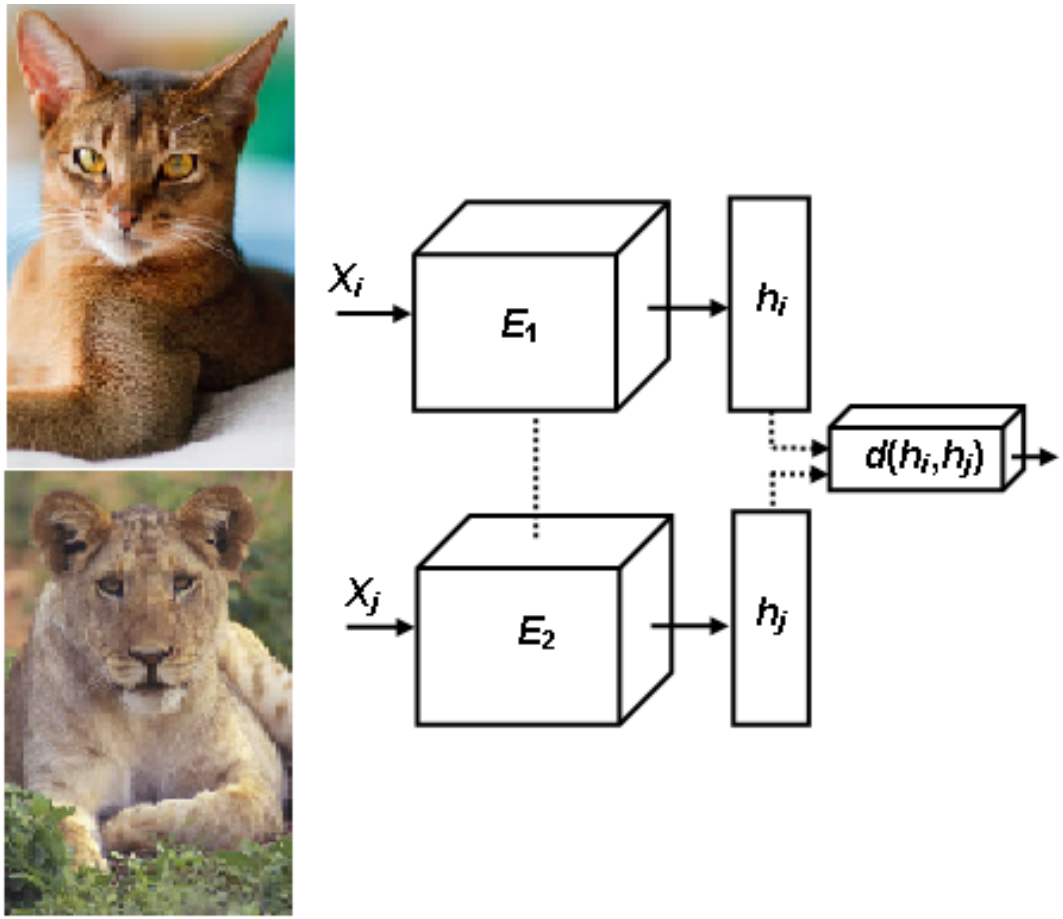
- **Weak-supervised case**
- Learning goal: to learn a metric that assigns small (large) distance to pairs of examples that are semantically similar (dissimilar) or to find a feature representation such that similar (dissimilar) objects are close to (far from) each other

Semantic similarity

Pets

Predators

Siamese neural network



How to implement Siamese NN by using DF?

1. We train every tree by using the concatenation of two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ such that the class $y_{ij} \in \{0, 1\}$ is defined by the semantical similarity of $\mathbf{x}_i$ and $\mathbf{x}_j$. Trees are trained on the basis of two classes.
2. We assign weights to trees in order to get "good" class vectors. The weights are computed in an optimal way in order to reduce distances between similar pairs and to increase them between dissimilar points.

Utkin L.V. & Ryabinin M.A. A Siamese Deep Forest. *Knowledge-Based Systems*, 2017 (In press)

Siamese deep forest (SDF)

- Condition for elements $v_{ij,0}$ and $v_{ij,1}$ of class vectors:

$$l(y_{ij}, v_{ij,0}, v_{ij,1}) = \left[\max\left(0, (2y_{ij} - 1)\left(v_{ij,0} - v_{ij,1}\right)\right)\right]^2.$$

- If $y_{ij} = 0$ ($y_{ij} = 1$), then we decrease (increase) the difference $v_{ij,1}^{(k)} - v_{ij,0}^{(k)}$
- The loss function ($z_{ij} = 2y_{ij} - 1$)

$$J_q(\mathbf{w}) = \sum_{i,j}\left[\max\left(0, \sum_{t=1}^{T} z_{ij}\left(p_{ij,0}^{(t)} - p_{ij,1}^{(t)}\right) w^{(t)}\right)\right]^2 + \lambda \left\|\mathbf{w}\right\|^2.$$

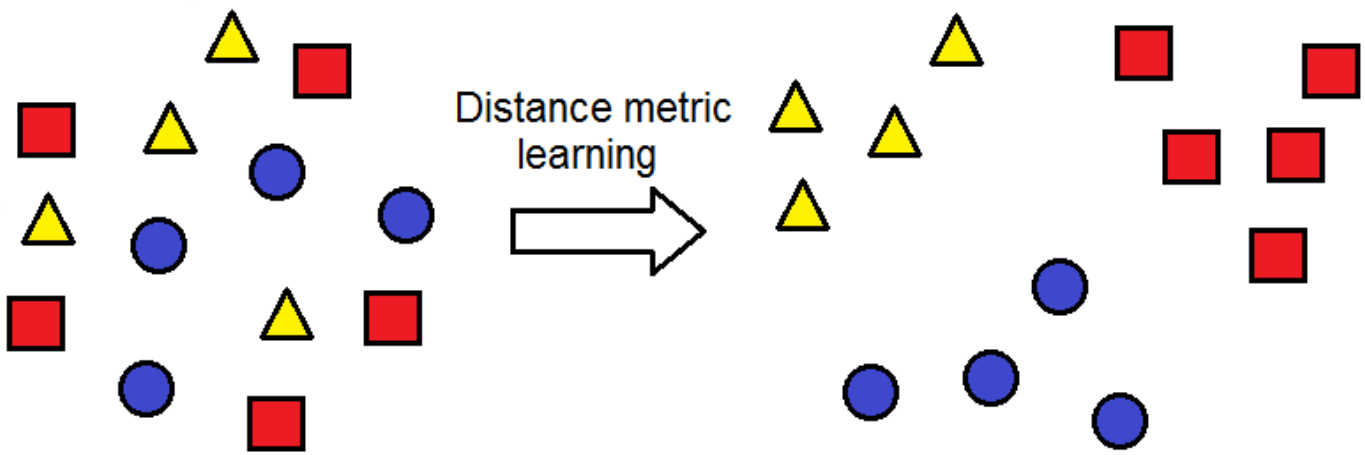
Siamese deep forest (experiments)

Table: Dependence of the accuracy measures on the number of pairs N and on the number of trees T in every forest for the MNIST data set

T	100		700		1000	
N	gcF	SDF	gcF	SDF	gcF	SDF
100	**0.470**	**0.490**	**0.570**	**0.585**	**0.530**	**0.570**
500	0.725	0.735	0.695	0.700	0.670	0.670
1000	0.757	0.770	0.775	0.780	0.830	0.840

Discriminative Deep Forest (DisDF)

- In contrast to the SDF, we have the **fully supervised learning** when the class labels of individual training examples are known



Discriminative Deep Forest (DisDF)

- The contrastive loss

$$l(\mathbf{x}_i, \mathbf{x}_j, y_i, y_j, \mathbf{w}) = ((1 - z_{ij})d(\mathbf{x}_i, \mathbf{x}_j) + z_{ij} \max(0, \tau - d(\mathbf{x}_i, \mathbf{x}_j))).$$

- In order to get the convex objective function we use **the Euclidean distance** in the first term and **the Manhattan distance** in the second term
- Finally, we get the quadratic optimization problem
- Utkin L.V., Ryabinin M.A. Discriminative Metric Learning with Deep Forest // *arXiv:1705.09620*, 25 May 2017.

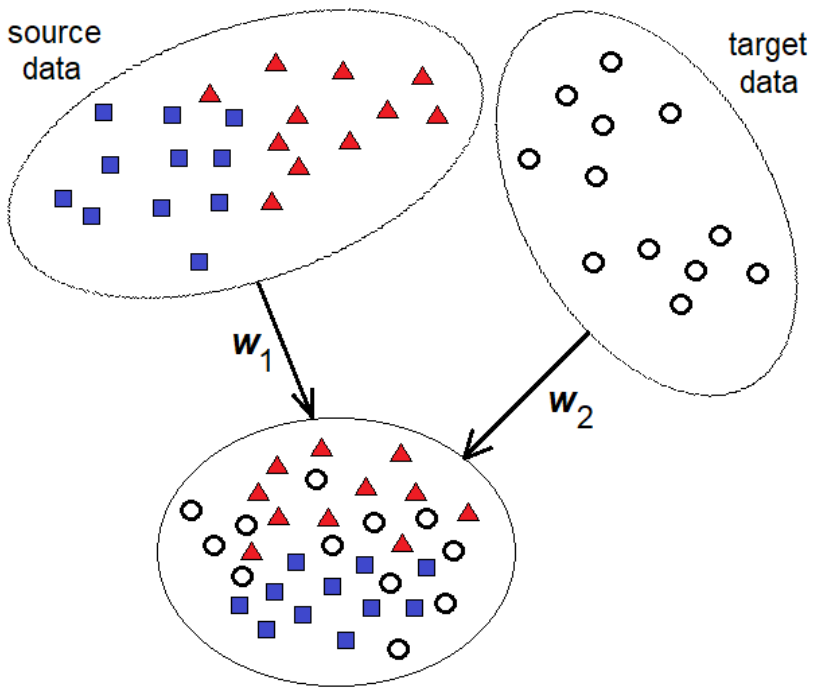
Transfer learning Deep Forest (TLDF)

Transfer learning problem statement

- The labeled *source domain data:*
  $\mathcal{D}^S = \{(\mathbf{x}_j, y_j),\ j = 1, ..., n_S\}$, $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \in \mathcal{X}^S$,
  $y_j \in \{1, 2, ..., C\}$.
- The unlabeled *target domain data:*
  $\mathcal{D}^T = \{(\mathbf{z}_j),\ j = 1, ..., n_T\}$, $\mathbf{z}_j$ is the j-th example.
- Transductive transfer learning or *domain adaptation*.
- Feature spaces between the source and target domain are the same, but the marginal probability distributions of data are different.
- We aim to train a classifier to make precise predictions on the target domain data $\mathcal{D}^T$.

Transfer Learning Problem Solution

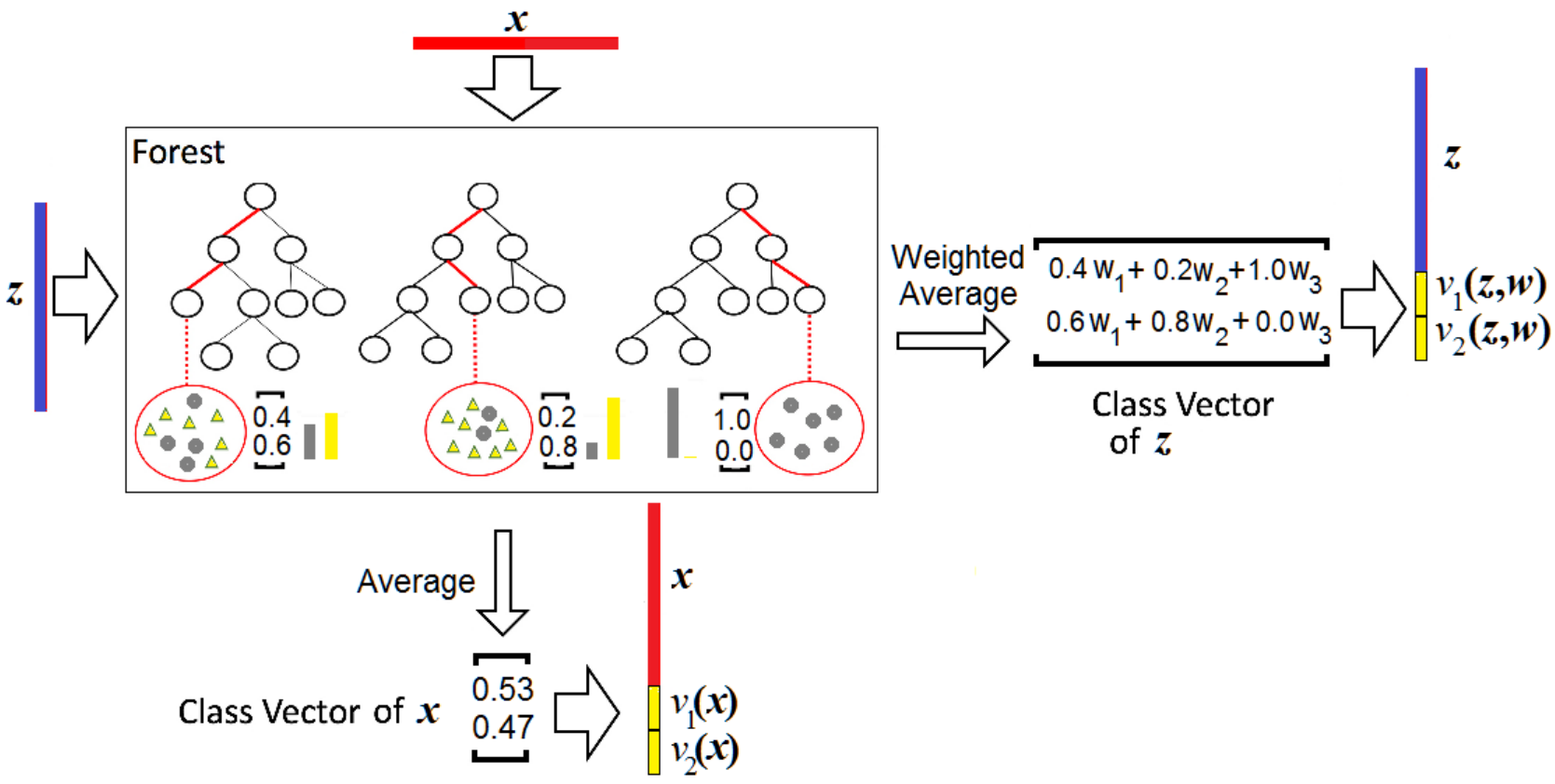
- One of the ways is the source and target domains alignment in a feature space
- A kind of the alignment is the consensus transfer learning



A general algorithm for one level

1. Trees are trained by using source data
2. By using the trained trees, we classify examples $\mathbf{z}$ from the target data
3. The class vector of $\mathbf{z}$ is computed as the weighted average
4. As a result, we get the augmented features $v_c^{(k)}(\mathbf{z}, \mathbf{w}^{(k)})$, $c = 1, ..., C$
5. The elements $v_c^{(k)}(\mathbf{z}, \mathbf{w}^{(k)})$ form the k-th forest class probability distribution $V^{(k)}(\mathbf{z}, \mathbf{w}^{(k)})$
6. Concatenated vector $\mathbf{z}$:
$\mathbf{z} \leftarrow \left(\mathbf{z}, V^{(1)}(\mathbf{z}, \mathbf{w}^{(1)}), ..., V^{(M)}(\mathbf{z}, \mathbf{w}^{(M)})\right)$

Weighted sums of domains



How to find weights?

- We have to maximize the consensus measure $C(V(\mathbf{z}, \mathbf{w}))$ or to minimize the Shannon entropy $H(V(\mathbf{z}, \mathbf{w}))$ over $\mathbf{w}$

$$H(V(\mathbf{z}, \mathbf{w})) = -\sum_{c=1}^{C} \left(\mathbf{p}_c^{(k)}(\mathbf{z})\mathbf{w}^{(k)}\right) \log \left(\mathbf{p}_c^{(k)}(\mathbf{z})\mathbf{w}^{(k)}\right) \rightarrow \min_{\mathbf{w}^{(k)}}$$

- The quadratic optimization problem:

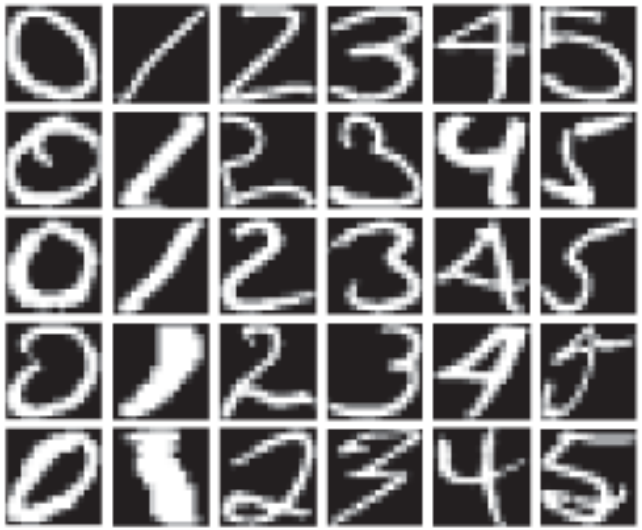
$$\min_{\mathbf{w}^{(k)}} \sum_{\mathbf{z} \in \mathcal{D}^T} \beta(\mathbf{z}) + \lambda \left\| \mathbf{w}^{(k)} \right\|^2$$

subject to $\mathbf{w}^{(k)}\mathbf{1}_k^{\mathrm{T}} = 1$ and

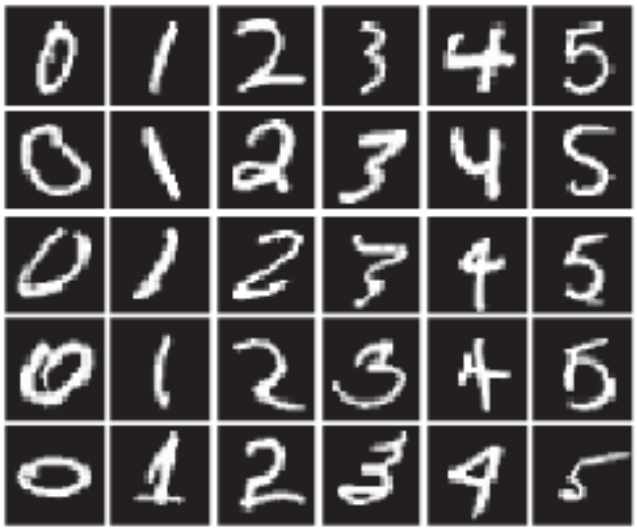
$$\beta(\mathbf{z}) + 2\mathbf{p}_j^{(k)}(\mathbf{z})\mathbf{w}^{(k)} \geq 2, \ \forall j \in \{1, ..., C\}.$$

Numerical experiments (1)

Two well-known public datasets USPS and MNIST



USPS



MNIST

Numerical experiments (2)

	ATTM	JAE	CDDA-a	CDDA-b	**TLDF**
MNIST vs USPS	77.94	87.6	76.22	82.33	79.1
USPS vs MNIST	61.15	86.9	62.05	70.75	73.4

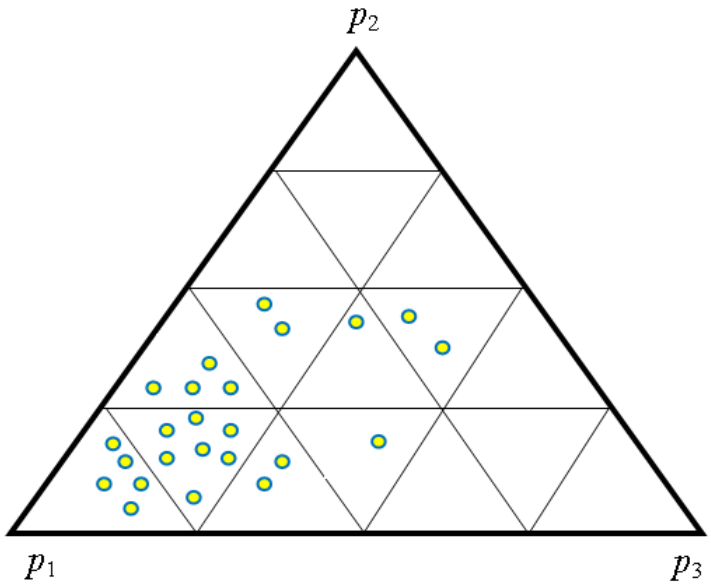
ATTM (Farajidavar et al., 2014), JAE (Epstein et al., 2017), CDDA-a and CDDA-b (Luo et al., 2017)

How to reduce the number of weights?

Problems of many weights

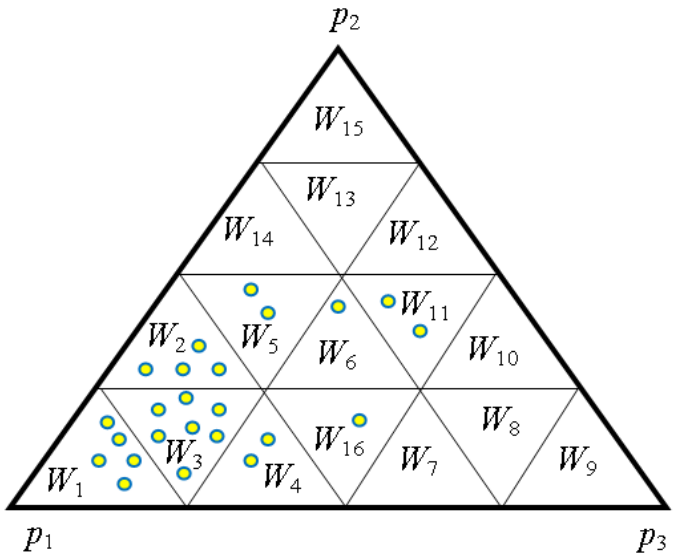
- The Deep Forest was developed in order to significantly reduce the number of training parameters which take place in neural networks
- The IDF again trains a lot of weights (the number of trees) as training parameters
- We get rid of many parameters in neural network and become many parameters in IDF
- We can reduce the number of trees in every random forest, but this leads to worse results and this is not a good way

How to reduce the number of weights?



- Yellow points are class prob. distrib. of all trees for $\mathbf{x}_i$
- Divide the unit simplex of probabilities of classes into $M = S^{C-1}$ small simplices
- Divide all trees for $\mathbf{x}_i$ into M subsets such that their class prob. distrib. are close to each other

How to reduce the number of weights?

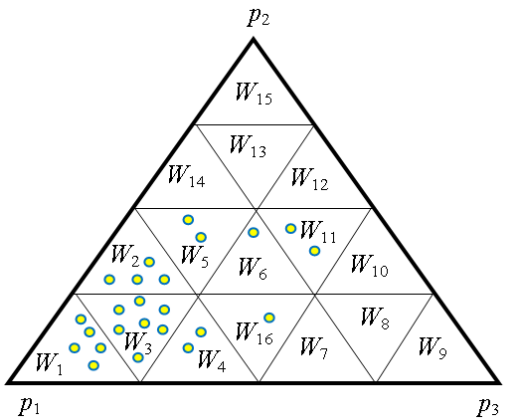


- **We assign weights not to trees, but to small simplices or to subsets of the class prob. distributions!**
- Condition for the small simplex $\Delta(j_k, k)$:

$$\frac{j_k - 1}{S} \leq p_k \leq \frac{j_k}{S}, \; j_k = 1, ..., S, \; k = 1, ..., C-1,$$

$$p_C = 1 - (p_1 + ... + p_{C-1})$$

The loss function for classification



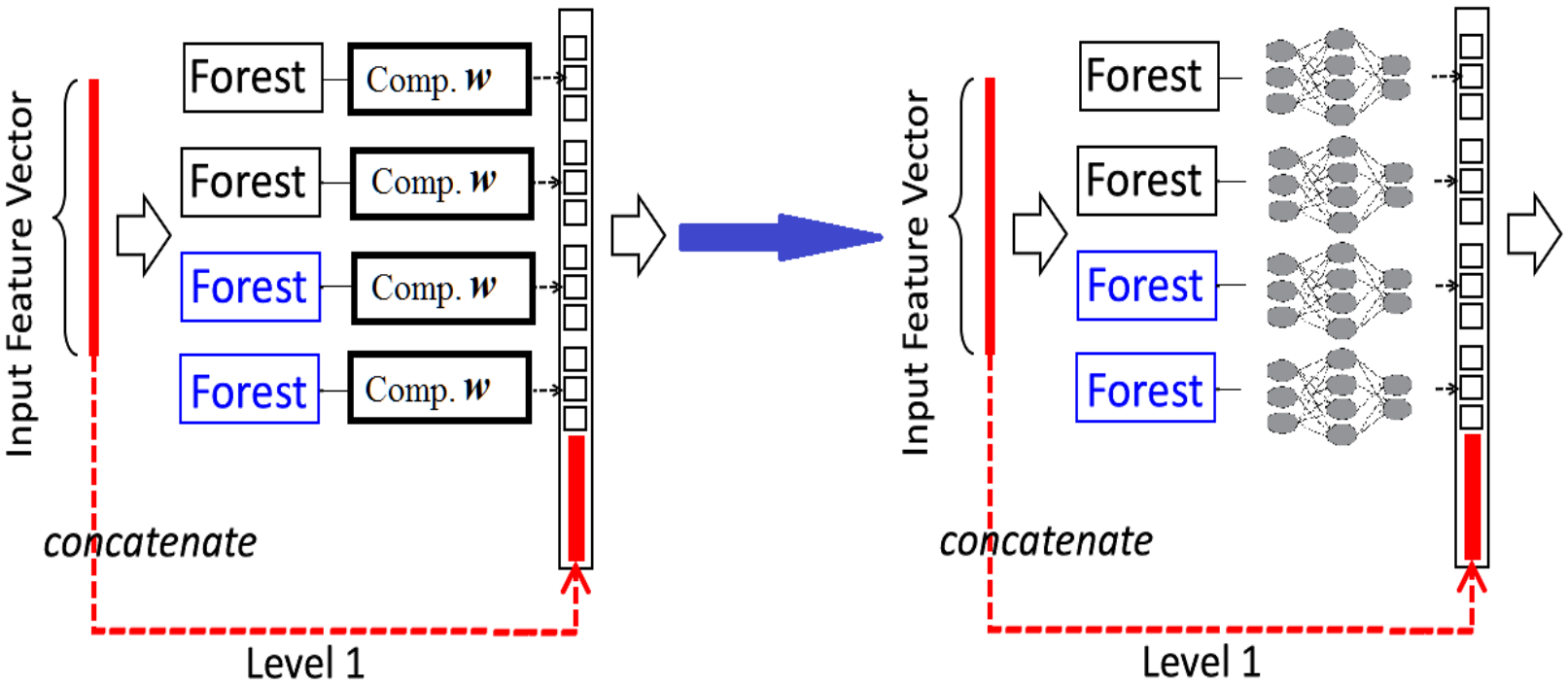
$$J(\mathbf{w}) = \min_{\mathbf{w}} \sum_{i=1}^{N} \sum_{c=1}^{C} \sum_{k=1}^{M} \left(W_k \sum_{t=1}^{T} p_{i,c}^{(t)} \mathbf{1}\left(p_{i,c}^{(t)} \in \Delta(j_k, k) \right) - \mathbf{1}(y_i = c) \right)^2 + \lambda \left\| W \right\|^2$$

subject to $\sum_{k=1}^{M} W_k = 1$

Another weight model

- The first extreme case: the Deep Forest - weights are identical, i.e. a point in the T-dimensional unit simplex
- The second extreme points: the IDF - weights are arbitrary in the T-dimensional unit simplex
- We propose to reduce the set of weights by applying the imprecise statistical models (the linear-vacuous mixture or imprecise ε-contaminated model, the pari-mutuel model, the constant odds-ratio model)
- All these models simply modify the Frank-Wolf algorithm

Deep Neural Forests (Deep NeuroForest)



Conclusion

1. The Deep Forest is a powerfull tool for classification especially when there are no much data
2. We have now a tool for controlling the deep forest like neural networks
3. Many machine learning problems can be solved by using the proposed weighted average
4. Many applications can be implemented by means of the proposed tool.

Questions

?