A Machine-learning Approach to Scene Classification and Flux Estimation using CERES-only TOA Radiances

Bijoy Vengasseril Thampi[1], Takmeng Wong[2]

Constantine Lukashin[2]

[1]Science System Applications Inc., Hampton, VA

[2]NASA Langley Research Center, Hampton, VA

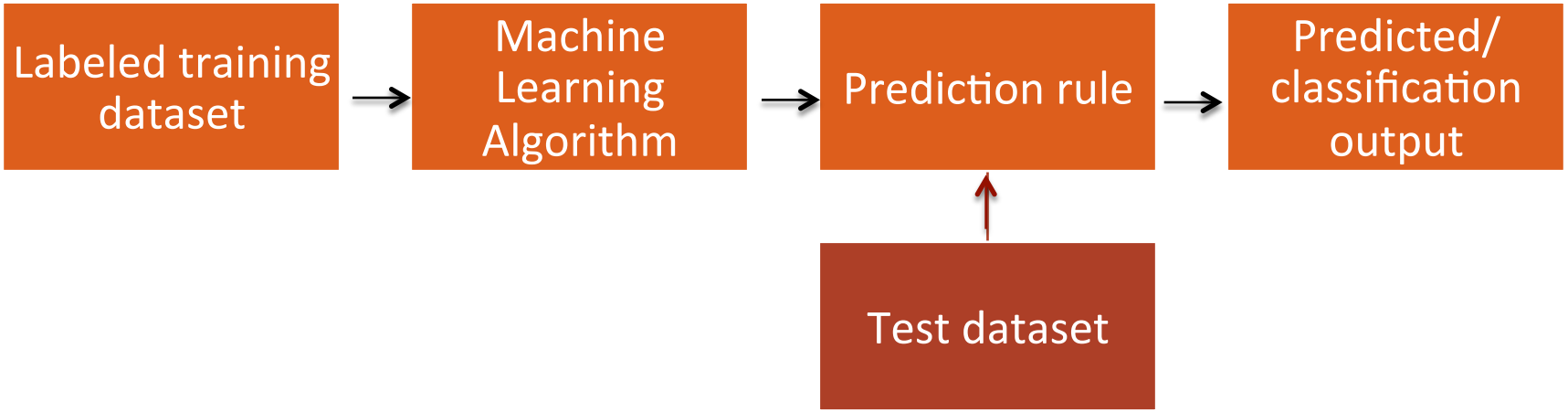
CERES Science Team meeting

University of Washington, Seattle, 1-3 September 2015

Objective

The objective is to develop a Machine learning methodology for the improved determination of scene type and subsequent TOA Flux estimation of CERES TOA radiance measurements devoid of Imager coverage.

Machine Learning Algorithms



- Scene Classification - Random Forests method
  - Developed by Breiman and Cutler(2000)
- TOA Flux estimation – Artificial Neural network method
  - ANN methodology discussed in the Lukashin and Loeb(2003)

Random Forests

- ***Random Forests (RF)***, is an ***ensemble learning*** method for classification and regression.
- RF use **decision tree classifiers** as the base model.
- Random forests operate by constructing a multitude of decision trees and outputting the class that gets maximum number of votes from the forest.

Main advantages of RF method are

- they have faster runtimes
- they can deal with unbalanced and missing data
- ability to handle data without preprocessing or rescaling.

Random Forests- Training data

Input variables are selected for the scene classification are:

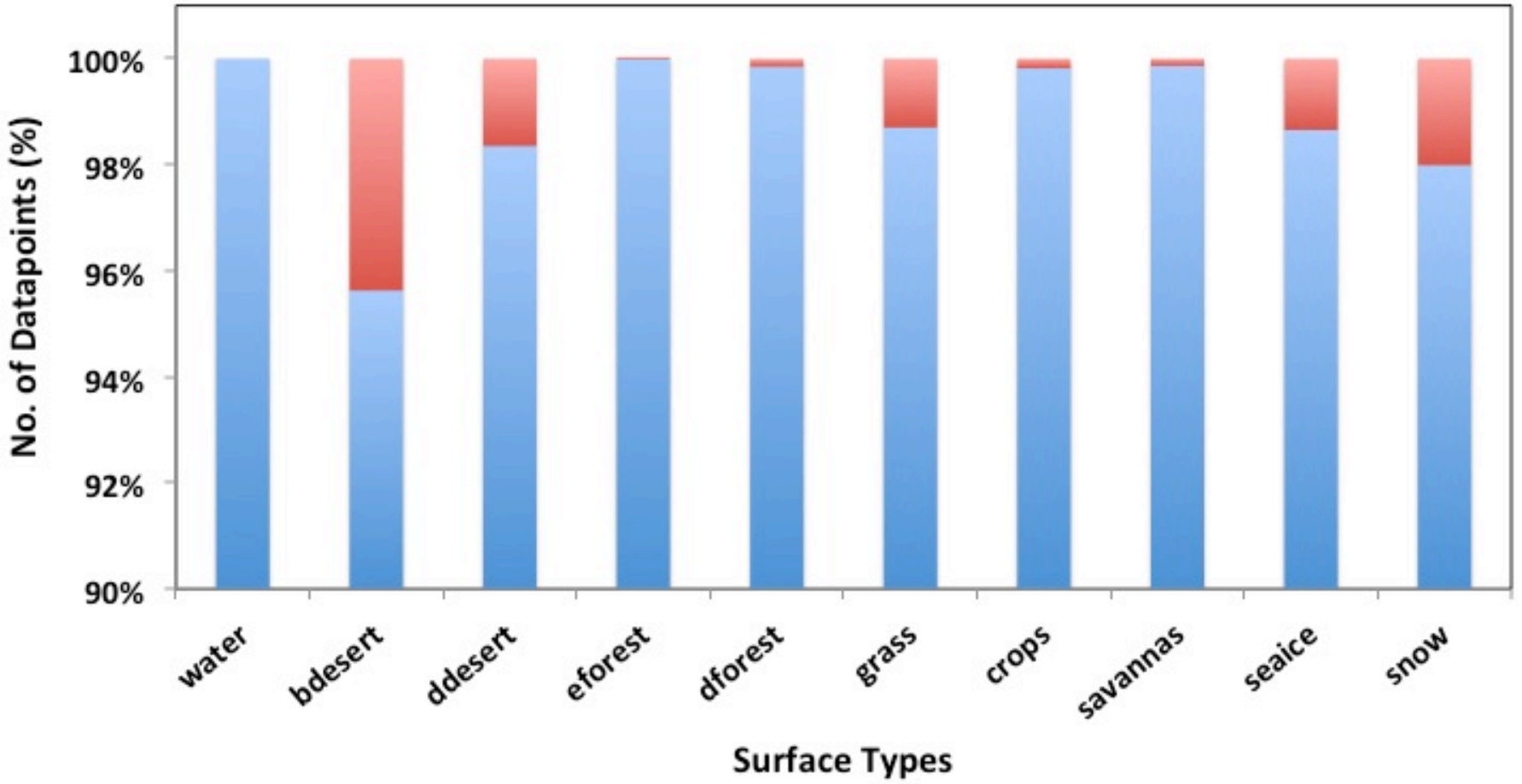
CERES	Ancillary data
Solar zenith & viewing zenith angles Relative azimuth angle CERES LW & SW broadband radiances IGBP Surface type	LW surface emissivity Broadband surface albedo Surface skin temperature Precipitable water Wind speed

IGBP Surface Types	
Water bodies	Evergreen Forests
Bright Desert	Deciduous Forests
Dark Desert	Woody Savannas and Shrub lands
Grasslands	Permanent and Fresh snow
Croplands and cities	Sea Ice

RF Results

Month : July (Day time)

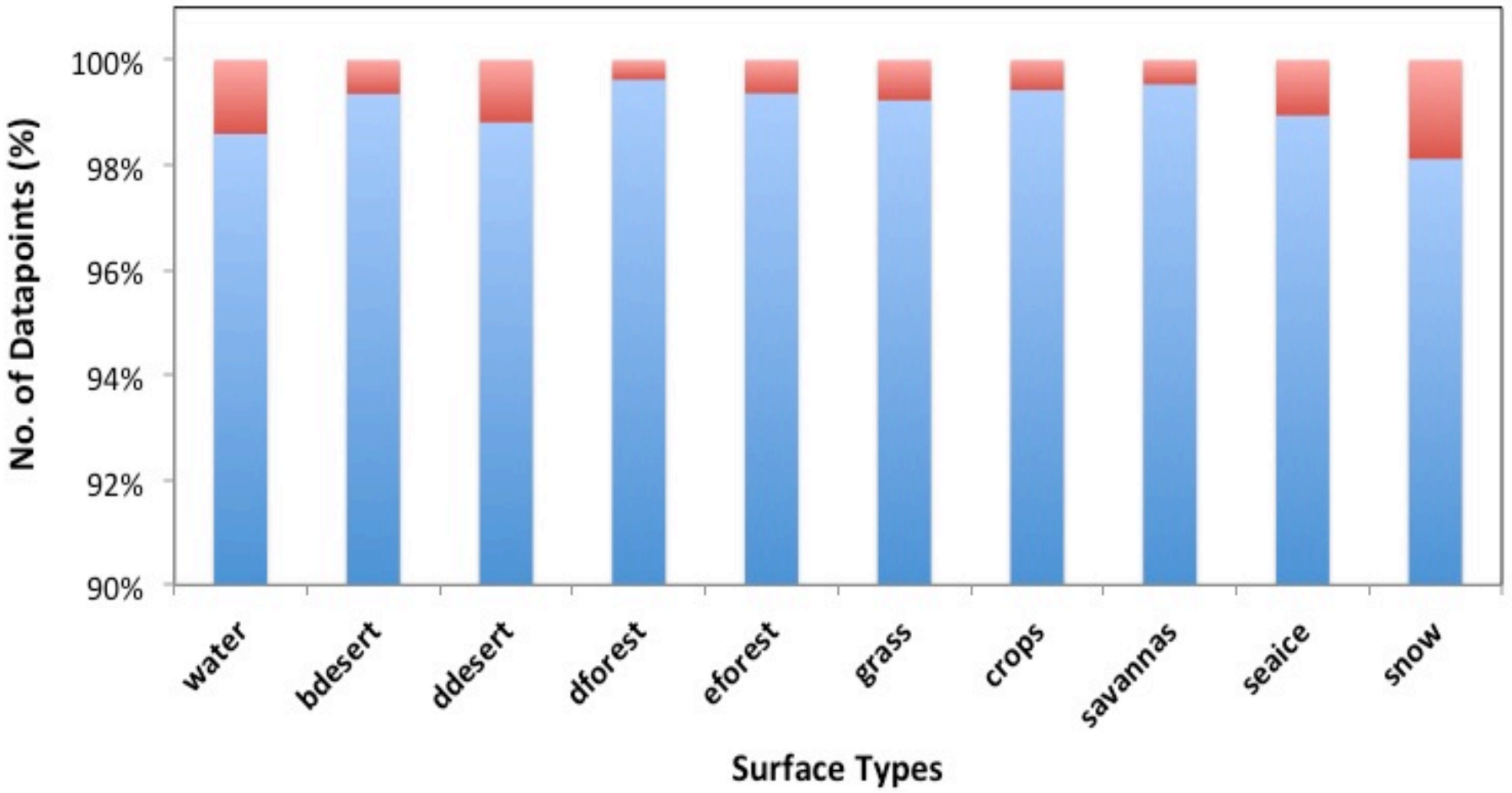
RED – misclassified data points



RF Results

Month : July (Night time)

RED – misclassified data points



TOA Flux estimation- 'ANN allsky'

- Once the Scene identification is carried out by RF, corresponding radiance to Flux conversion can be carried out by employing a feed-forward error back-propagation (FFEB) artificial neural network (ANN) method (**Loukachine and Loeb, 2003**).
- ANN is used to relate the CERES SW and LW allsky radiances and viewing geometry directly to the SW and LW anisotropic correction factors.
- The technique is then validated by comparing ANN-derived TOA fluxes with SSF TOA fluxes.
- The training dataset used in this ANN computation encompasses both clear and cloudy sky data (full year)..

We decided to try a new approach for **ANN ANALYSIS**…

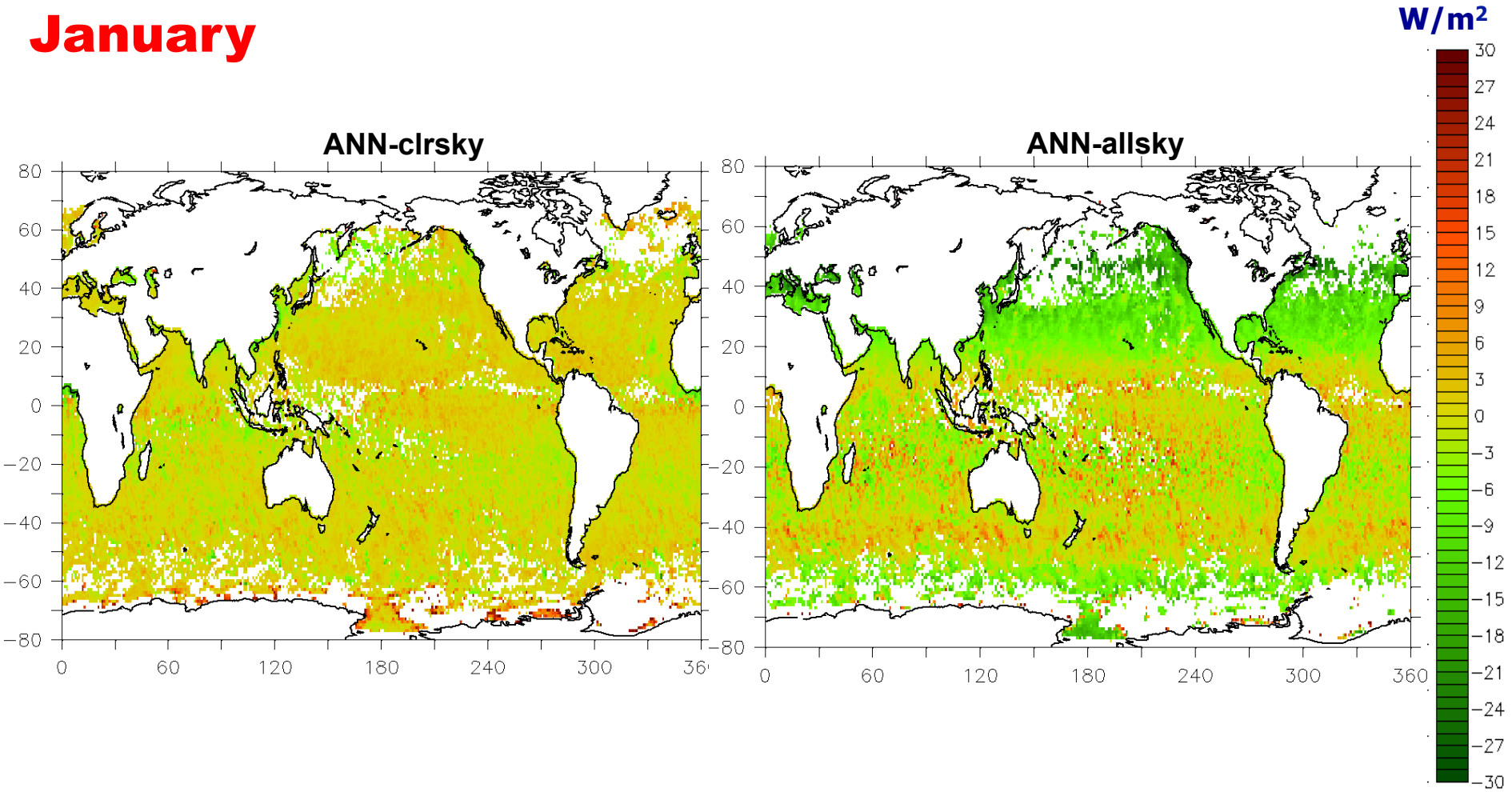
'ANN - clear sky' approach

- The modified approach use only CERES clear sky SW, LW and WN radiances for a particular month spanning several years (Aqua, 2003 to 2012) as training dataset (input).
- Adopted the same training methodology followed by Loukachine and Loeb (2003) but only using clear sky data.
- Output TOA flux from the ANN-based model is compared with CERES Aqua SSF fluxes.
- **Test data -** January & July (Day time data only)

  Year - 2013 & 2014

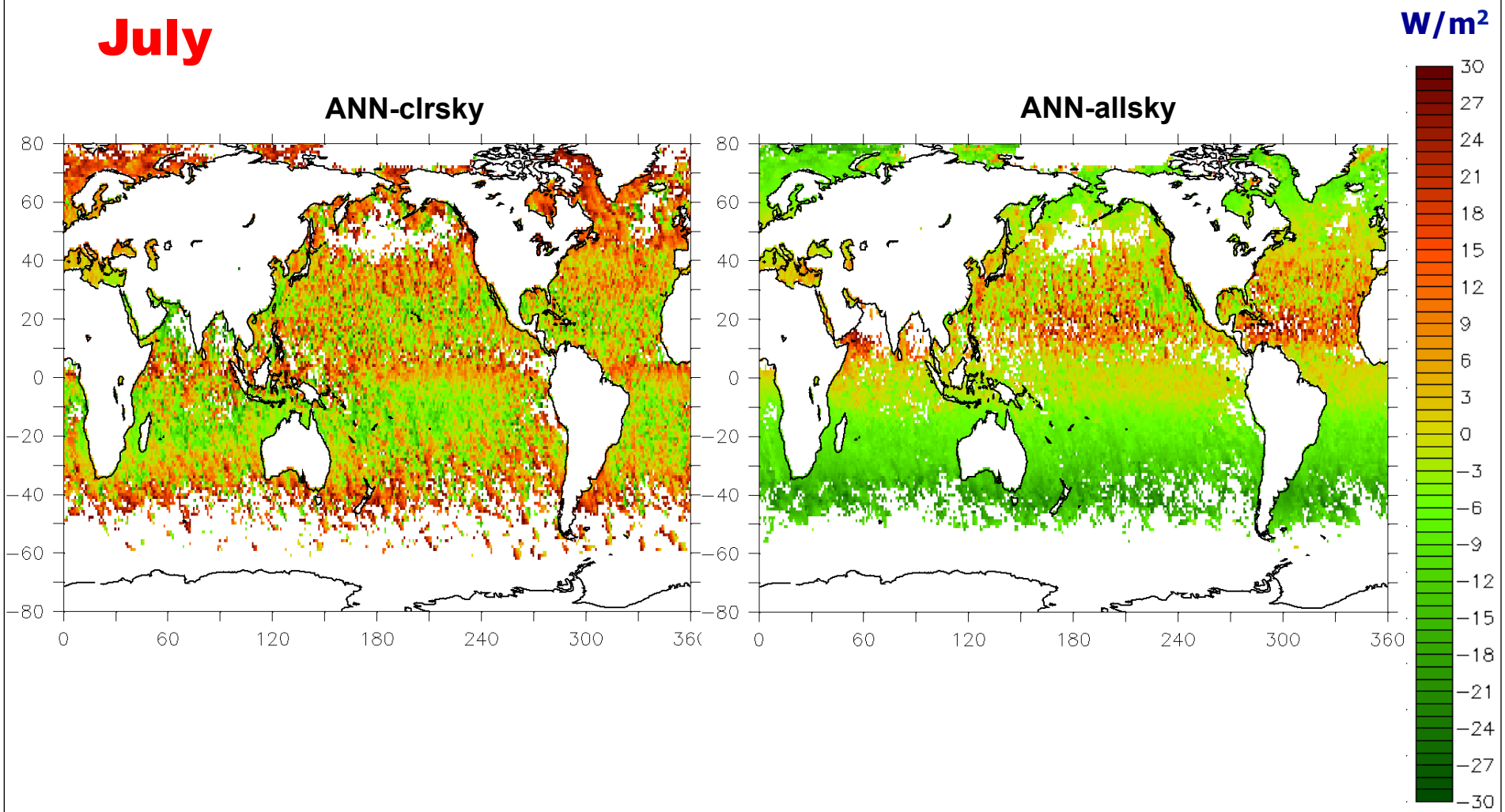
Difference between ANN-derived and CERES/Aqua TOA fluxes: SW

January



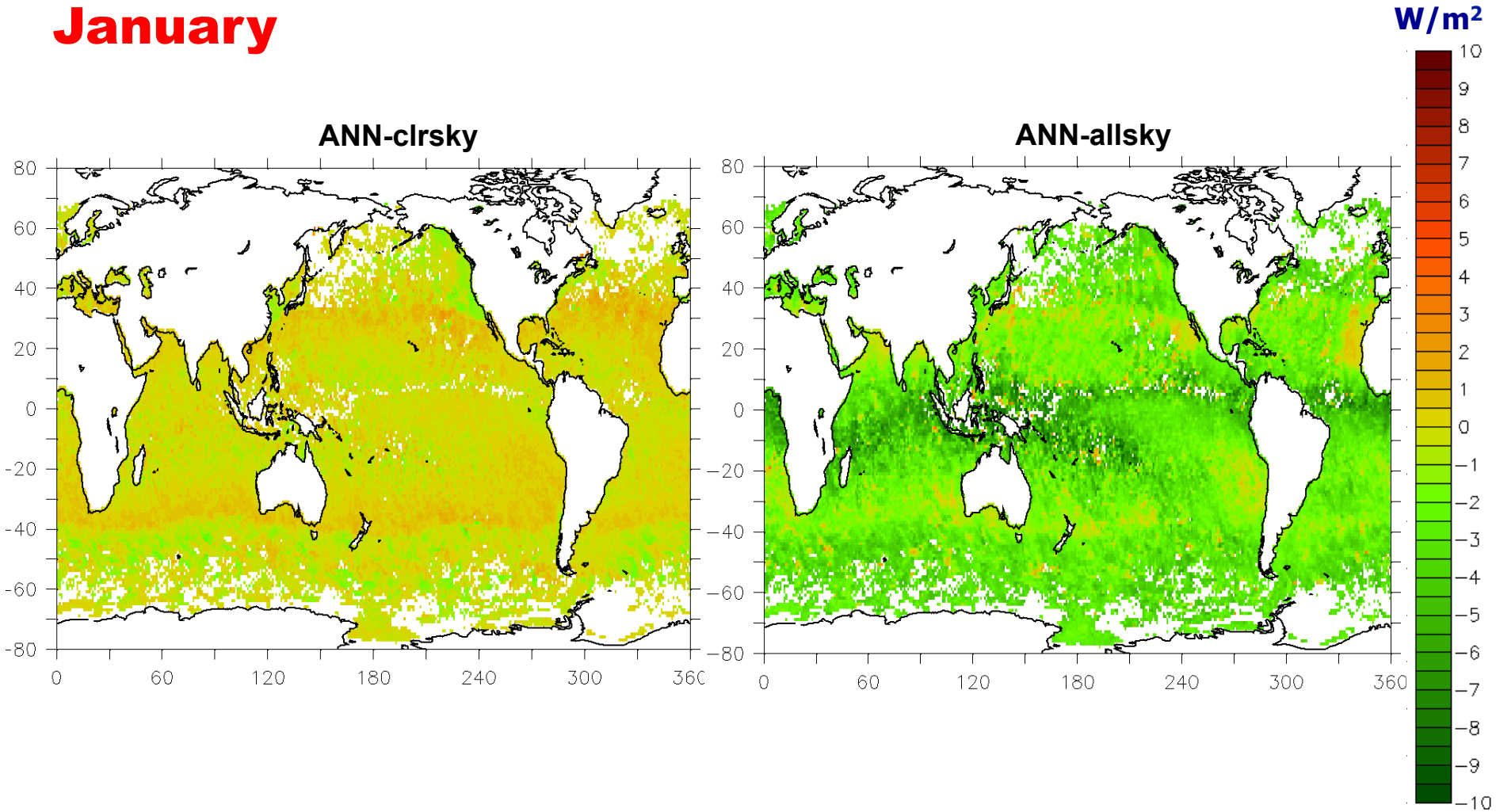
Difference between ANN-derived and CERES/Aqua TOA fluxes: SW

July



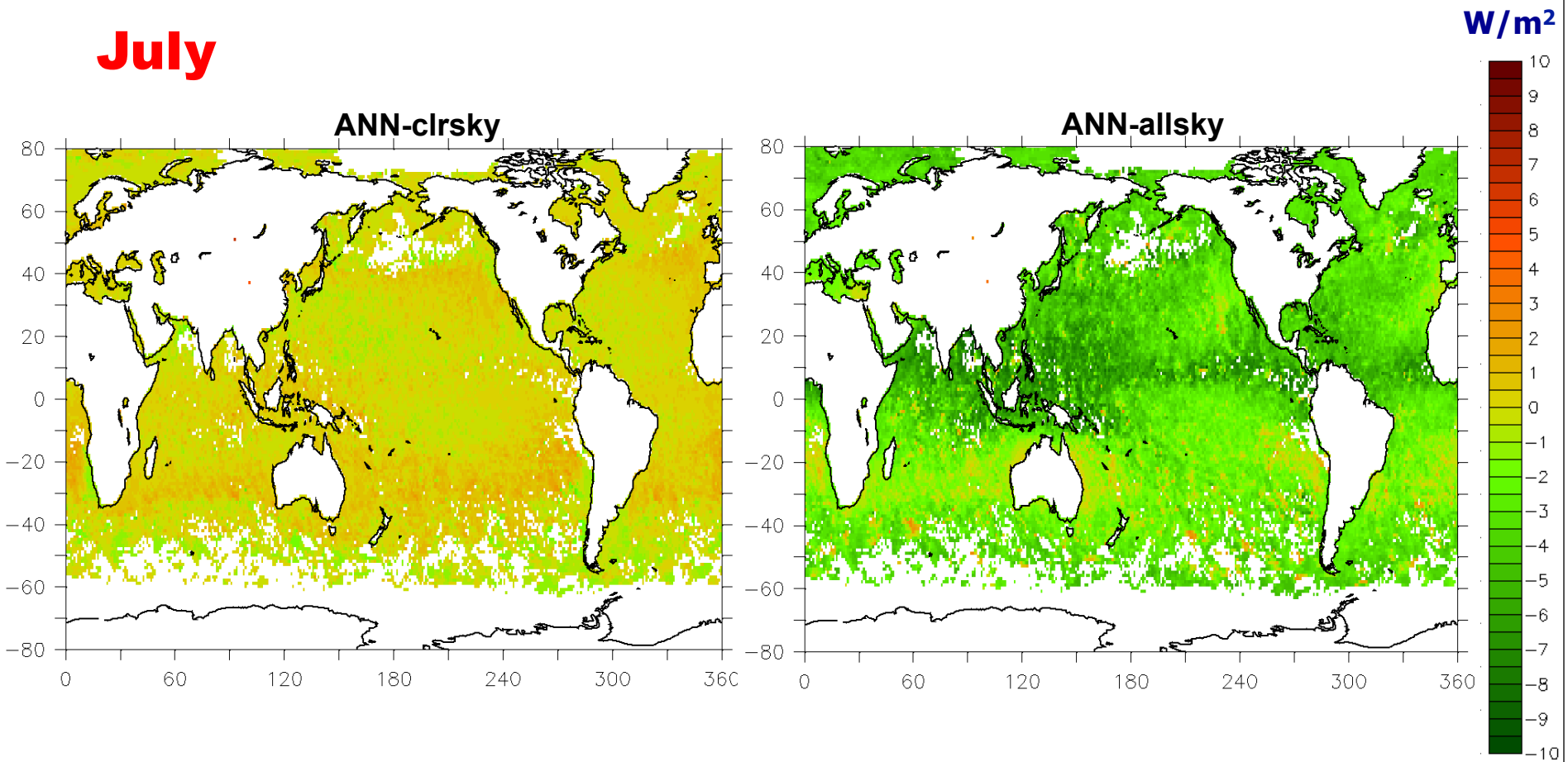
Difference between ANN-derived and CERES/Aqua TOA fluxes: LW

January



Difference between ANN-derived and CERES/Aqua TOA fluxes: LW

July



Global mean BIAS and RMS

BIAS = global mean of the difference between the ANN-derived and original CERES TOA fluxes

Month	TOA Flux	BIAS (%)		RMSD (%)	
		ANN-clrsky	ANN-allsky	ANN-clrsky	ANN-allsky
January	SW	0.28	-2.90	4.15	8.32
	LW	0.03	-0.80	0.30	1.16
	WN	0.02	-0.56	0.39	0.95
July	SW	0.53	-3.84	5.16	14.67
	LW	0.03	-0.99	0.29	1.34
	WN	0.03	-0.69	0.38	1.07

Summary

- Finished the RF analysis using day time and night time data.
- The Bias and RMSD values show relatively lower for the ANN-clrsky method.
- Based on the analysis using ~1.5 million clear sky test data points (over Ocean), ANN-clrsky approach for SW TOA flux produces lower bias values for ~60-70% of test cases.
- For LW and WN flux cases, observed test cases with relatively lower bias values using ANN-clrsky method is between ~85-90% and ~79-84% respectively.

Future Work: Finish the ANN-clrsky analysis for other surface types and months.

THANK YOU…..

Machine learning

Machine learning deals with the construction and study of systems that can learn from data. It focuses on prediction, based on known properties of the system learned from the training data.

Reduced variance: results are less dependant on peculiarities of a single training set.

Reduced bias : combination of multiple classifiers may produce more reliable classification than single classifier.

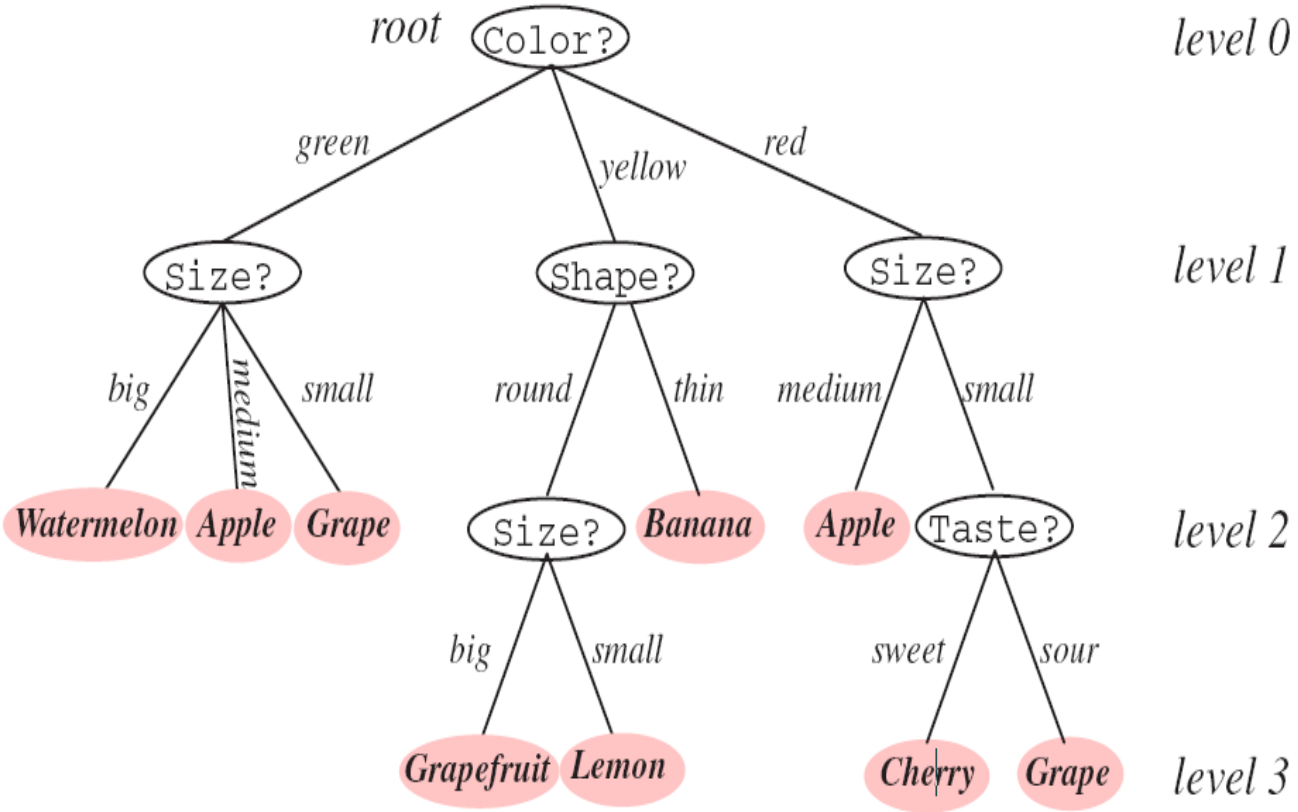
Eg., SVM, Bayes optimal classifier, Boosting, Bagging, Random forest

Random forests, first proposed by Tin Kam Ho of Bell Labs in 1995, is an ensemble learning method for classification and regression.

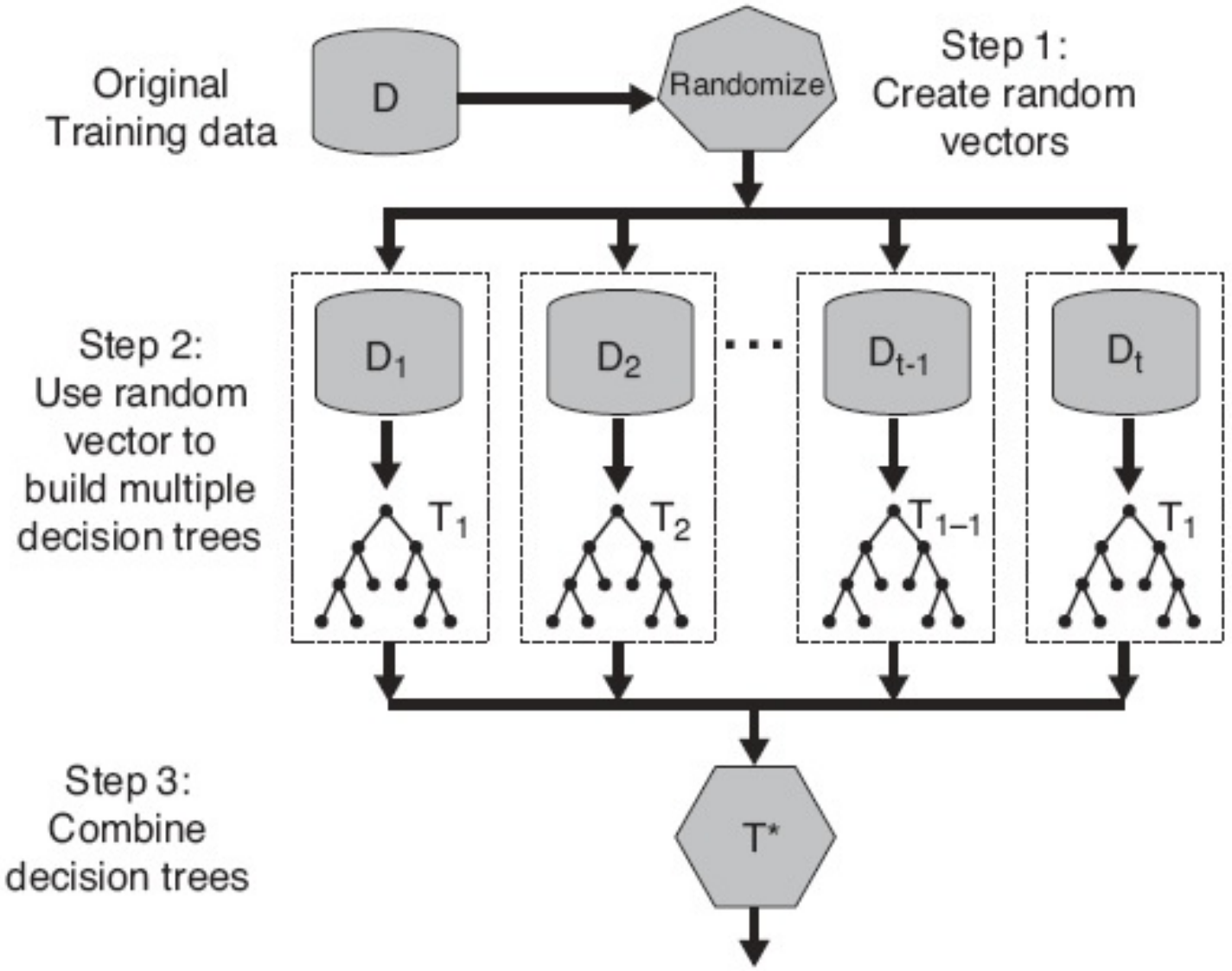
Decision Tree

Decision tree predictors are the basic unit of random forest predictors. Decision trees can express any function of the input attributes.

Good attribute: One which splits the examples into subsets that are (ideally) "all positive" or "all negative"



Random Forests –Flow diagram



Random Forest Algorithm

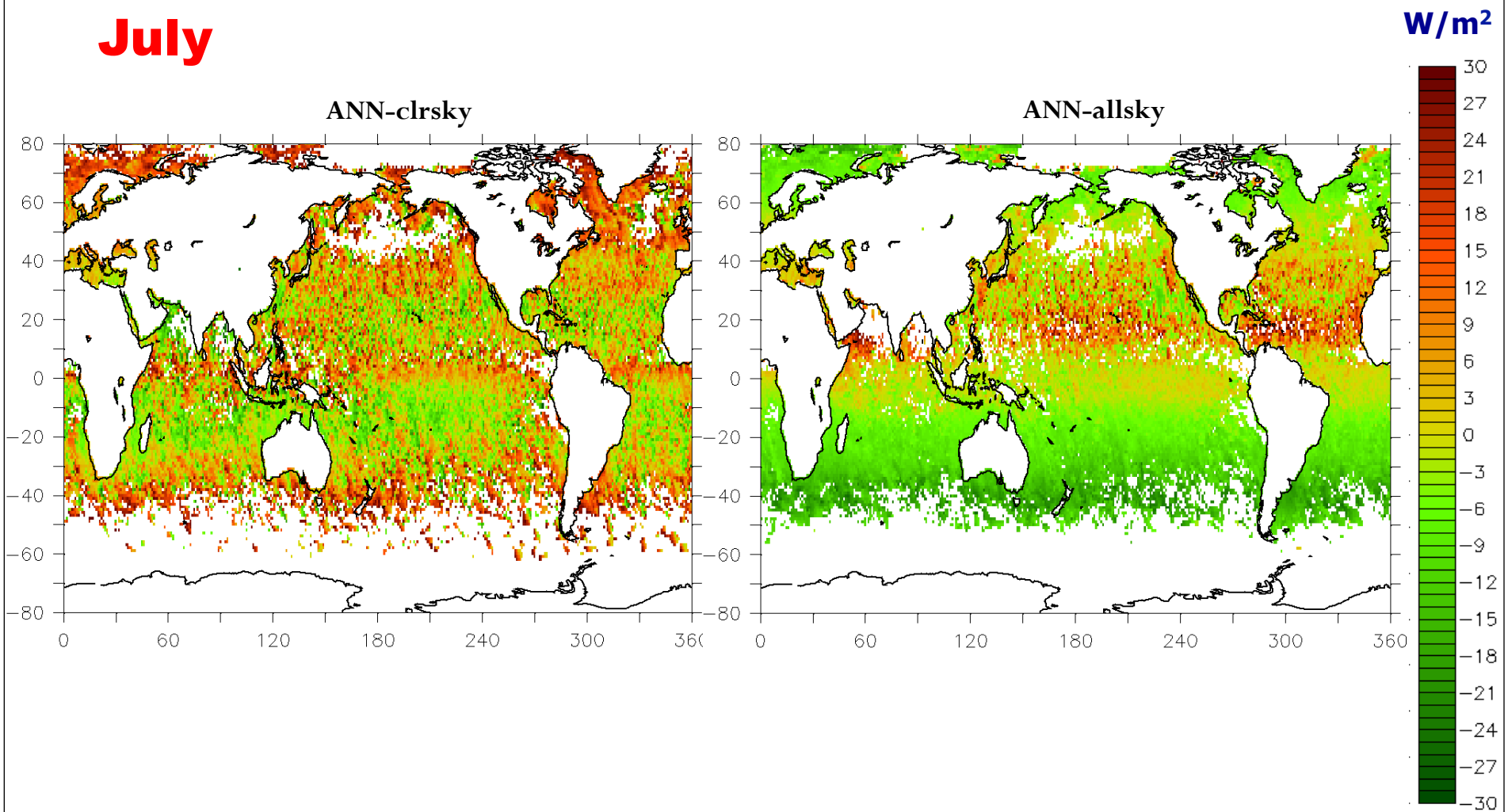
- Introduce two sources of randomness: "**Bagging**" and "**Random input vectors**".
- **Bagging**- repeated random sub-sampling of the training data.
- **Bootstrap sample** - will on average contain **63.2%** of the data while the rest are replicates.
- Using bootstrap sample, a decision tree is grown to its greatest depth determining the split at each node through minimizing the loss function.
- For each tree, using the leftover (**36.8%**) data, misclassification rate is calculated = **out of bag (OOB)** error rate.
- Aggregate error from all trees to determine **overall OOB error rate for the classification**

Artificial Neural network

- A **Neural Net (NN)** simulates the the learning functions of the brain. It can recognizes patterns and learn from them
- NN learns by determining the relation between the inputs and outputs
- This is done calculating the relative importance of inputs and outputs.
- Through trial and error, NN compares the results with expert provided values in the output until user defined accuracy level is reached.

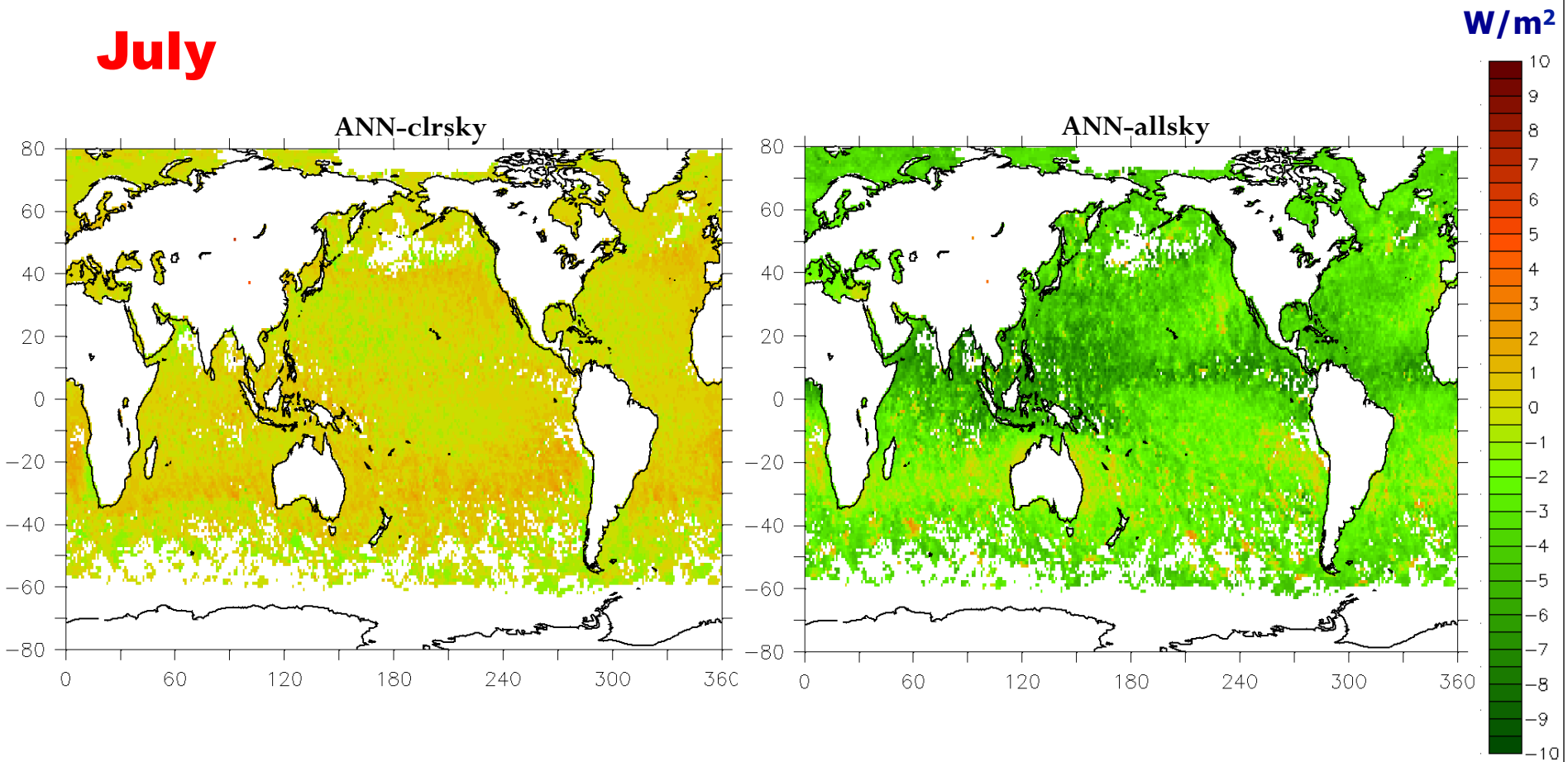
Regional difference between mean ANN-derived and CERES/Aqua TOA fluxes: SW

July

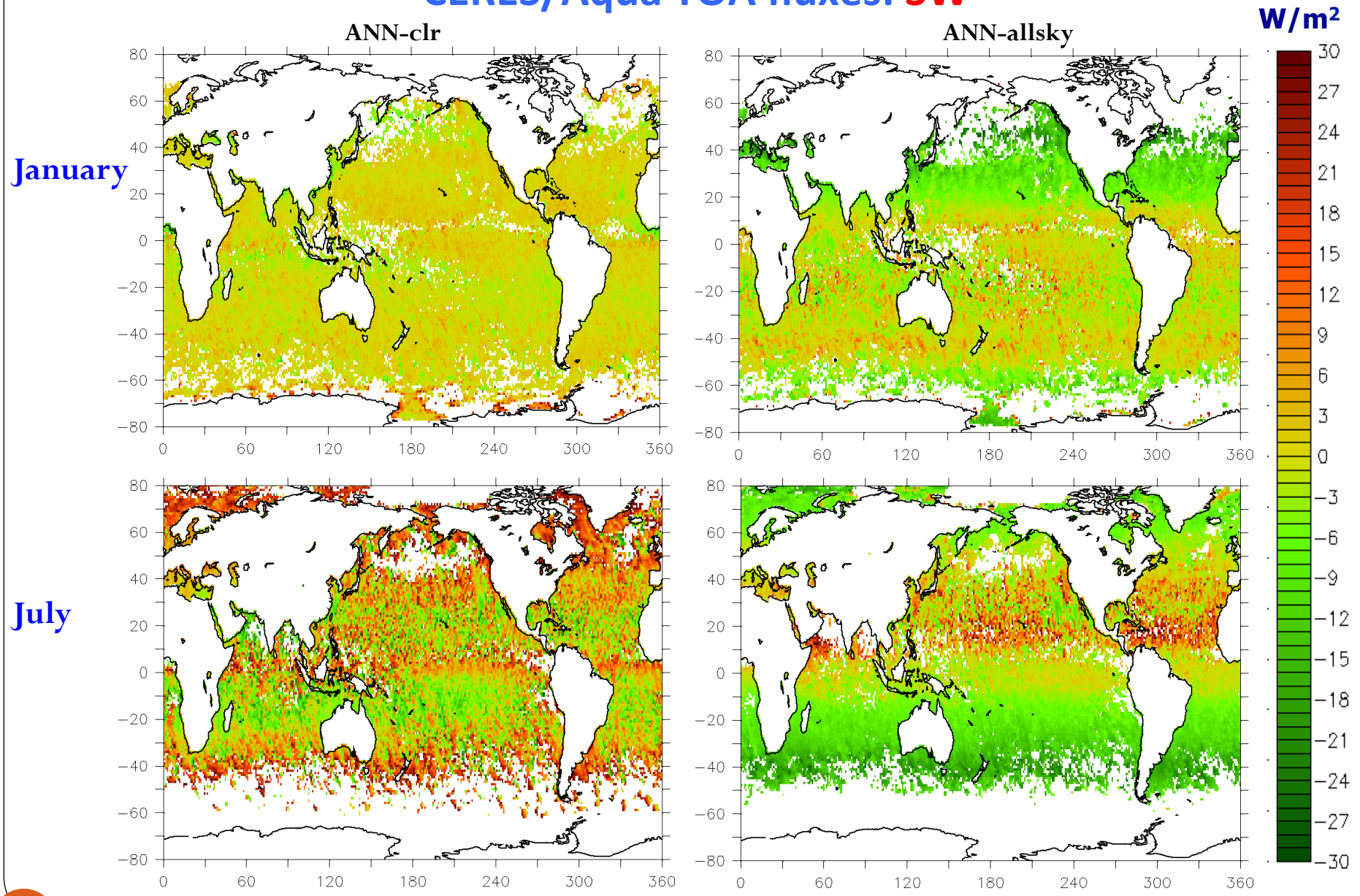


Regional difference between mean ANN-derived and CERES/Aqua TOA fluxes: LW

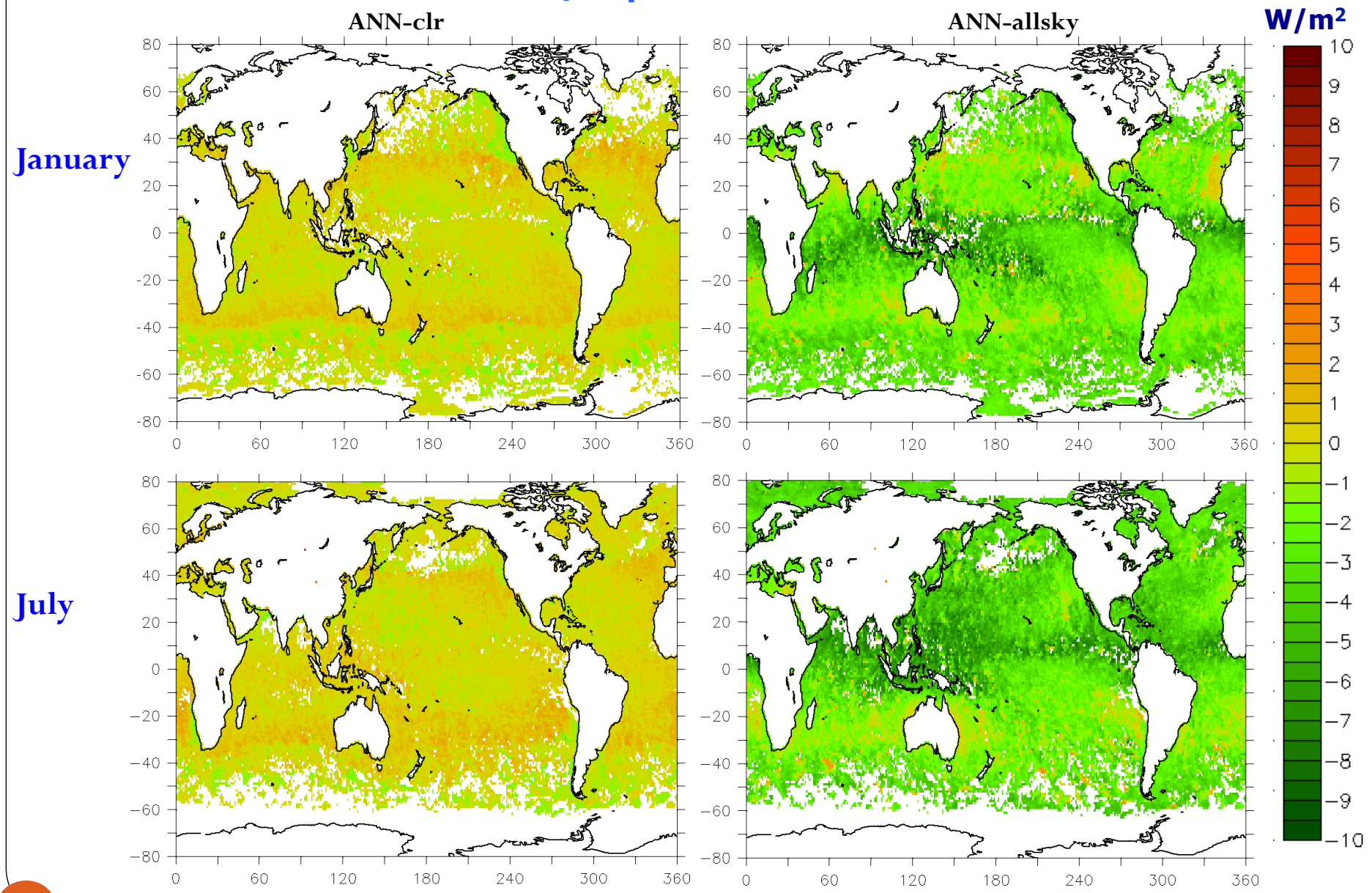
July



Regional difference between mean ANN-derived and CERES/Aqua TOA fluxes: SW



Regional difference between mean ANN-derived and CERES/Aqua TOA fluxes: LW



Global mean BIAS and RMS

BIAS = global mean of the difference between the ANN-derived and original CERES TOA fluxes (W/m^2)

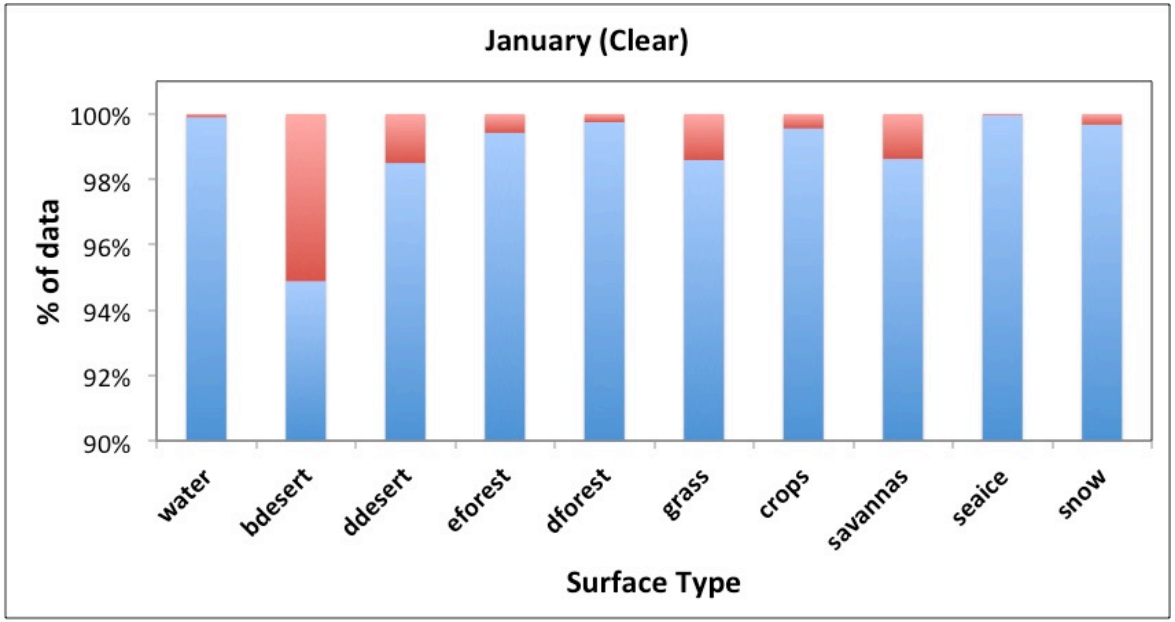
Month	TOA Flux	BIAS (W/m^2)		RMS (W/m^2)	
		ANN-clrsky	ANN-allsky	ANN-clrsky	ANN-allsky
January	SW	0.24	-2.48	3.54	7.09
	LW	0.08	-2.28	0.86	3.3
	WN	0.01	-0.49	0.35	0.84
July	SW	0.42	-3.08	4.05	11.76
	LW	0.09	-2.81	0.82	3.79
	WN	0.03	-0.60	0.33	0.93

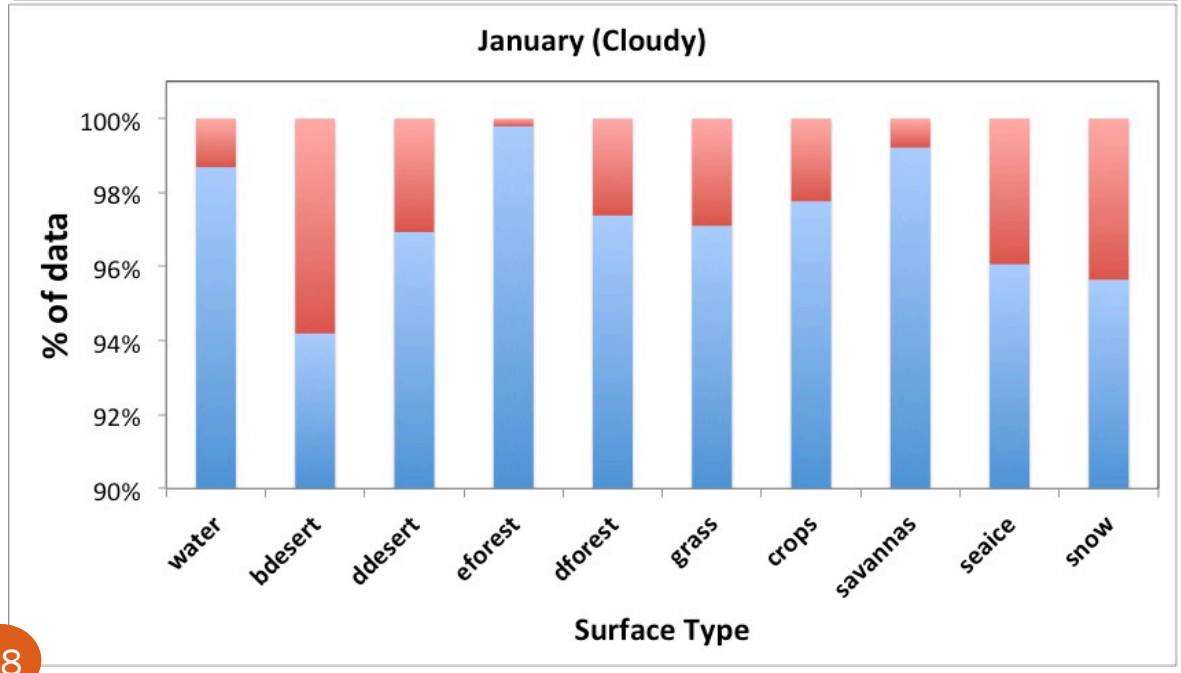
Global mean BIAS and RMS

BIAS = global mean of the difference between the ANN-derived and original CERES TOA fluxes (W/m^2)

		BIAS		RMS	
		ANNclr	ANNtot	ANNclr	ANNtot
January	SW	0.24 (0.28)	-2.48 (-2.90)	3.54 (4.15)	7.09 (8.32)
	LW	0.08 (0.03)	-2.28 (-0.80)	0.86 (0.30)	3.3 (1.16)
	WN	0.01 (0.02)	-0.49 (-0.56)	0.35 (0.39)	0.84 (0.95)
July	SW	0.42 (0.53)	-3.08 (-3.84)	4.05 (5.16)	11.76 (14.67)
	LW	0.09 (0.03)	-2.81 (-0.99)	0.82 (0.29)	3.79 (1.34)
	WN	0.03 (0.03)	-0.60 (-0.69)	0.33 (0.38)	0.93 (1.07)

RF Results- January (Day time)



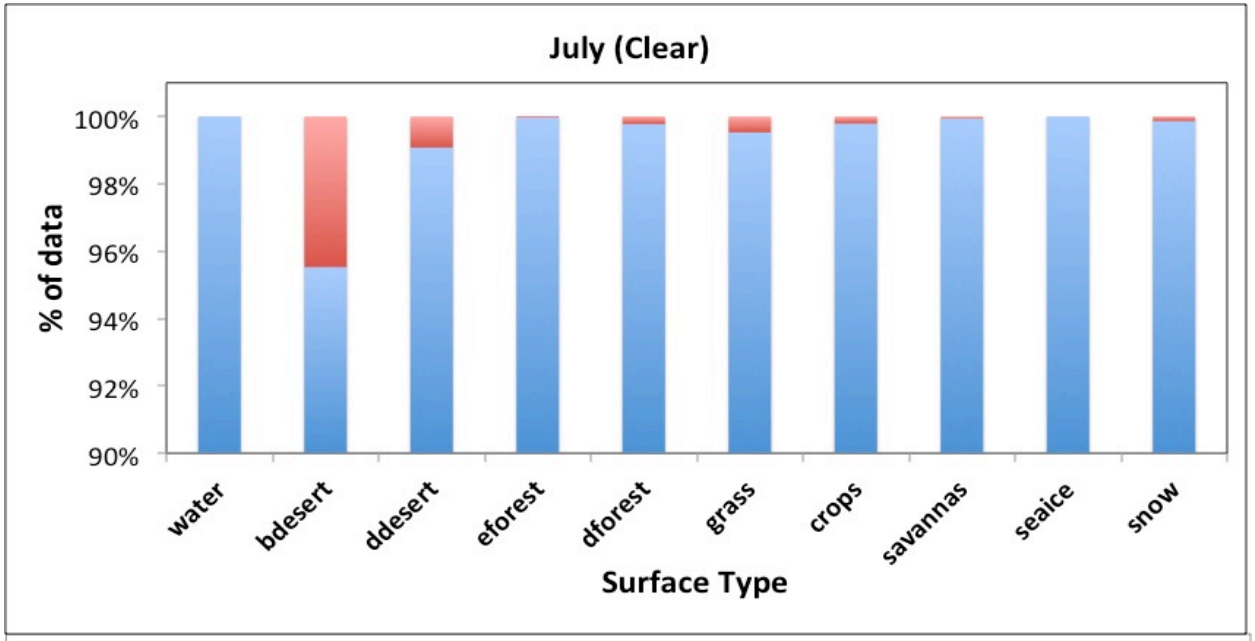


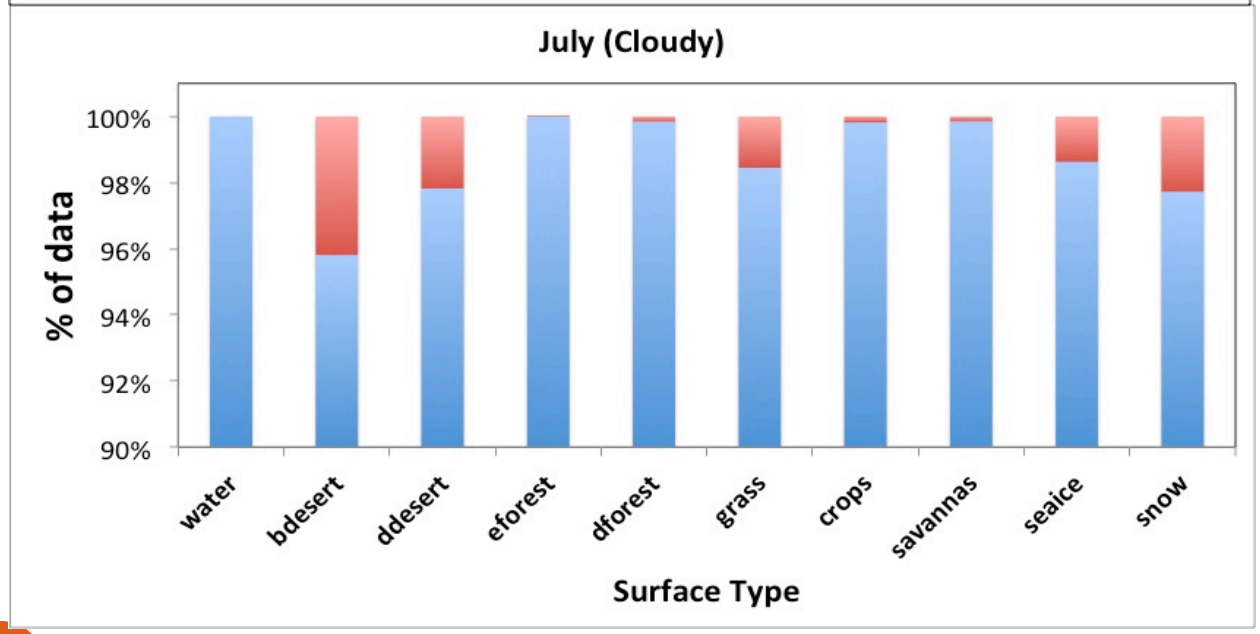
Month : January (Day time)

Relatively large misclassification rate was observed for surface types: Bright Deserts, Dark deserts, Grass lands and Snow

Lower misclassification rate was observed for surface types: Water bodies, Evergreen forest and Deciduous forests

RF Results

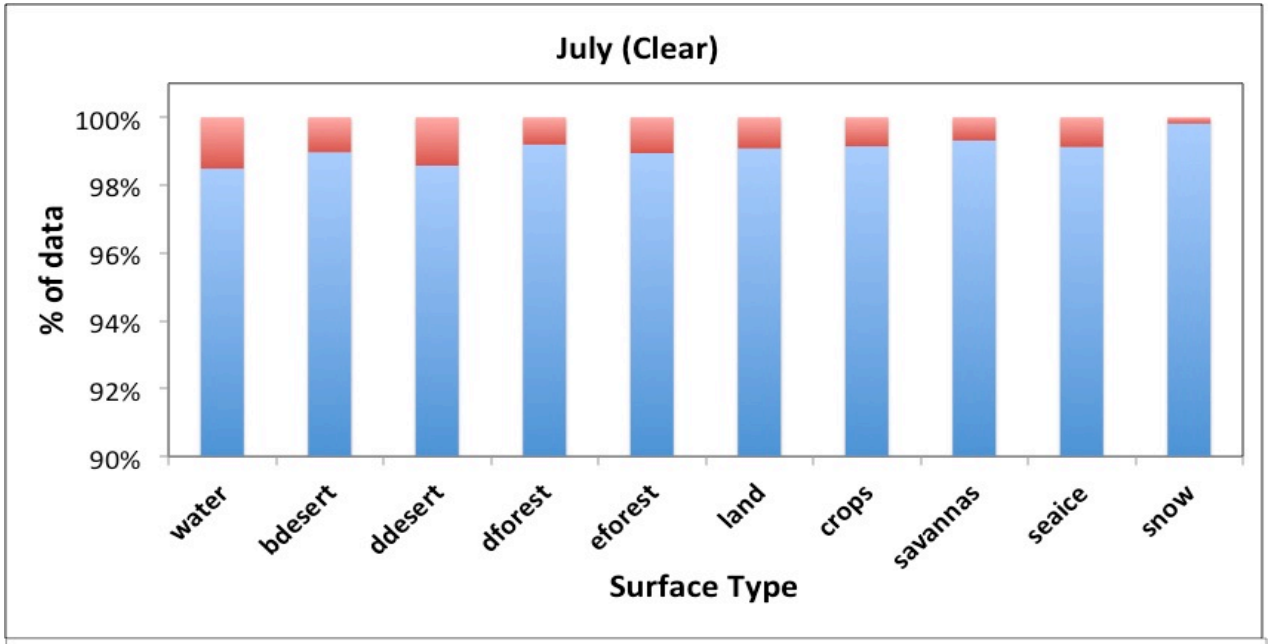


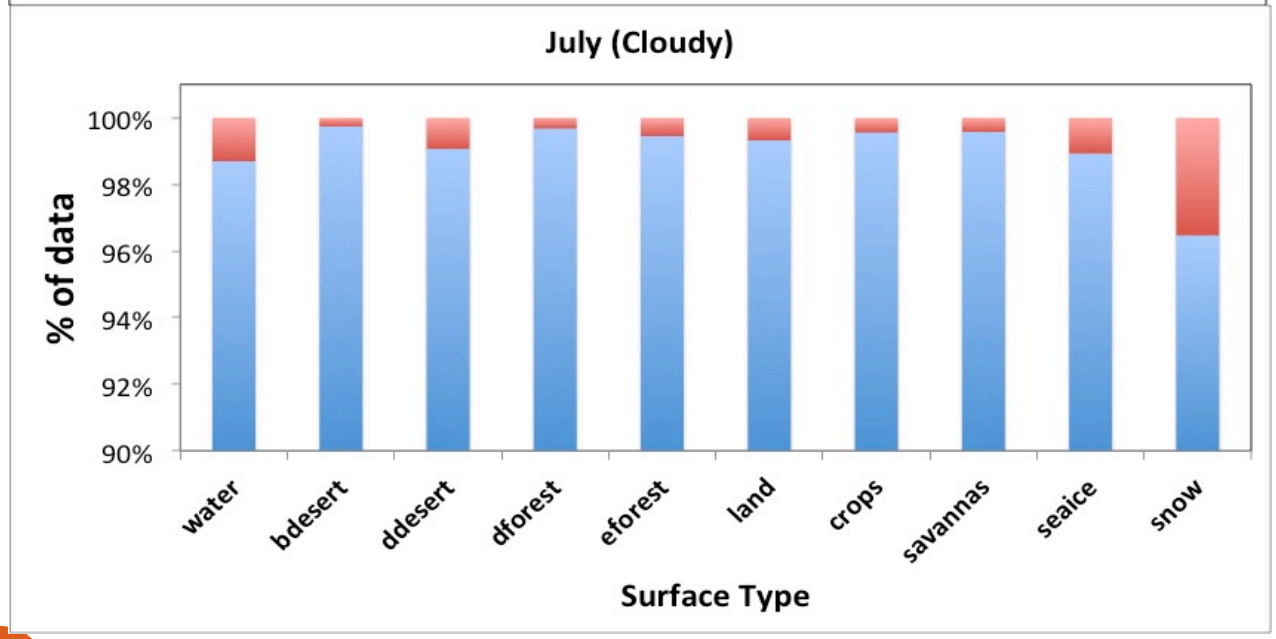


Month : July (Day time)

RED - Incorrectly classified data points

RF Results





Month : July (Night time)

RED - Incorrectly classified data points

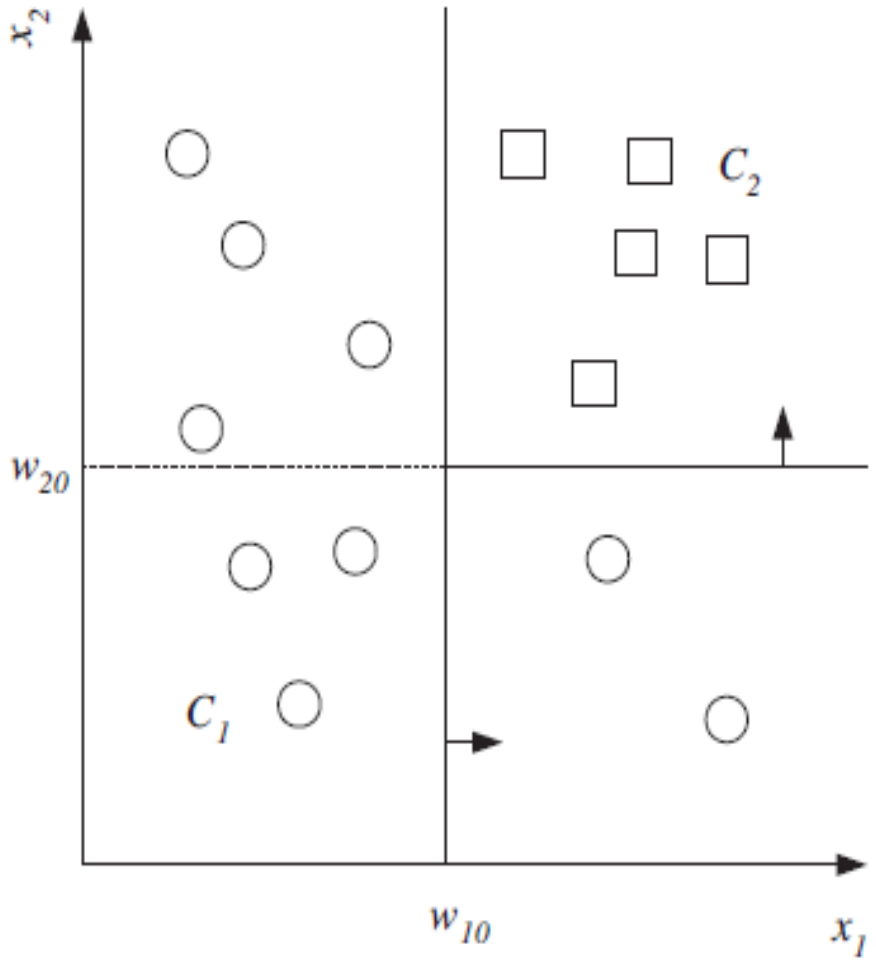
Random Forests

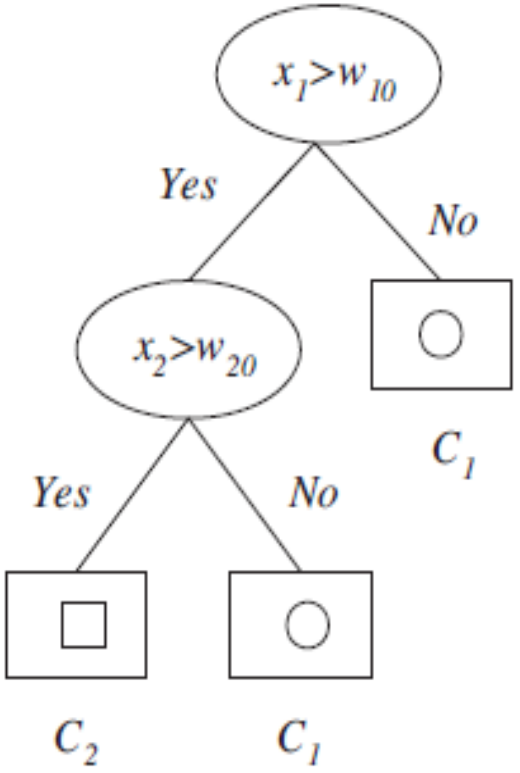
Introduction

- **Random forest** is an ensemble classifier that consists of many decision trees and outputs the class that is the mode of the class's output by individual trees.
- **Random forests** (RF) are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest.
- The generalization error of a forest of tree classifiers depends on the **strength** of the individual trees in the forest and the **correlation** between them.
- Using a random selection of features to split each node yields smaller error rates that compared to other ensemble methods

Decision Trees

- A **decision tree** is a hierarchical data structure implementing the divide and conquer strategy. It can be used for both classification and regression.
- A decision tree is composed of internal **decision nodes** and **terminal leaves.**
- A decision tree is a hierarchical model for supervised learning whereby the **local region is identified in a sequence of recursive splits.**
- The splitting process starts at the root and is repeated recursively until a **leaf node** is hit at which point the value written in the leaf constitutes the output.
- Mathematically speaking, each decision node (m) implements a test function (**fm(x)**) with discrete outcomes labeling the branches.





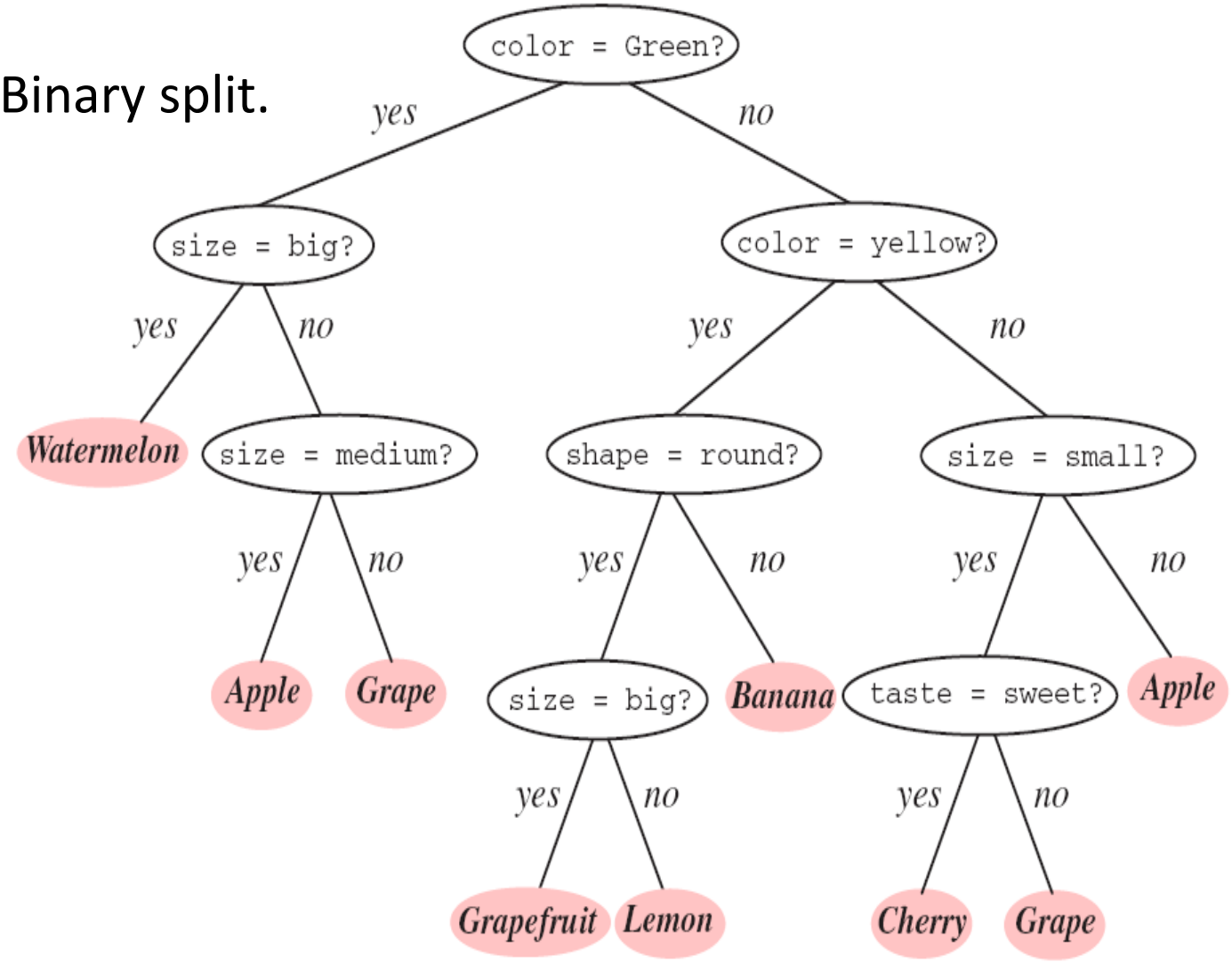
Example of a dataset and the corresponding decision tree. Oval nodes are the decision nodes and rectangles are leaf nodes. The univariate decision node splits along one axis,

Decision trees

- **fm(x)** defines a discriminant in the d-dimensional input space dividing it into smaller regions.
- fm(x) is a simple function and when written down as a tree, is broken down into a series of simple decisions. Different decision tree methods assume different models for fm(·), and the model class defines the shape of the discriminant and the shape of regions.
- Each leaf node has an output label, which in the case of **classification is the class code and in regression is a numeric value.**
- A leaf node defines a localized region in the input space where instances falling in this region have the same labels (in classification) or very similar numeric outputs (in regression).
- The hierarchical placement of decisions allows a fast localization of the region covering an input.
- Another advantage of the decision tree is interpretability. The tree can be converted to a set of **IF-THEN** rules that are easily understandable.

Classification Tree

Binary split.



Classification Tree

- For a **decision tree for classification** or a **classification tree**, the goodness of a split is quantified by an impurity measure.
- A split is pure if after the split, for all branches, all the instances choosing a branch belong to the same class.
- For node m, N_m is the number of training instances reaching node m. N_{im} of N_m belong to class Ci, with $\sum_i N_{im} = Nm$.
- Probability of class Ci, for an instance reaches node m, is gives an

$$P(Ci|x,m) \equiv p_{im} = N_{im} / N_m.$$

- Node m is pure if p_{im} for all i are either **0 or 1.** It is 0 when none of the instances reaching node m are of class Ci , and it is 1 if all such instances are of Ci .
- If the split is pure, we do not need to split any further and can add a leaf node labeled with the class for which p_{im} is 1.

Classification Tree

To decide what split criteria to use, need to establish the measurement of node impurity.

Entropy: $I_m = -\sum_i p_{im} \log_2 p_{im}$

Gini impurity: $\sum_i \sum_{j \neq} p_{im} p_{jm} = 1 - \sum_j p^2_{jm}$

Misclassification error: $1 - \max_i (p_{im})$

So for all attributes,we calculate the impurity and choose the one that has the minimum entropy. Then tree construction continues recursively and in parallel for all the Classification and branches that are not pure, until all are pure.

At each step during tree construction, we choose the split that causes the largest decrease in impurity, which is the difference between the impurity of data reaching node m and the total entropy of data reaching its branches after the split.

Methodology

Each **decision tree** in RF algorithm is constructed using the following logic:

1. Let the number of training cases be N, and the number of variables in the classifier be M.
2. ***m*** input variables are used to determine the decision at a node of the tree; m<<M and m>=1.
3. Choose a training set for this tree by choosing n times with replacement from all N available training cases (**Bootstrap sample**). Use the rest of the cases to estimate the error of the tree, by predicting their classes.
4. For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.
5. Each tree is fully grown and not pruned.

For prediction a new sample is pushed down the tree. It is assigned the label of the training sample in the terminal node it ends up in. This procedure is iterated over all trees in the ensemble, and the average vote of all trees is reported as random forest prediction.

Advantages

From Computational standpoint, RF are appealing because

- They can handle both regression and classification
- Relatively fast to train and predict
- Have a built in generalization error
- Can be used for high deimnsional problems
- Can be implemented easily

From Statistical standpoint, RF are appealing because of features like

- Measure variable importance
- Missing value imputation
- Outlier detection
- Unsupervised learning
- Intrinsic proximities between cases and Clustering
- Scaling coordinates based on the proximities

Variable importance

For the nth case in the data, its margin at the end of a run is the proportion of votes for its true class minus the maximum of the proportion of votes for each of the other classes.

To estimated the importance of the mth variable:

i) In the OOB cases for the kth tree, randomly permute all values of the mth variable

ii) Put these new variable values down the kth tree and get classifications.

iii) Compute the margin.

The measure of importance of the mth variable is the **average lowering of the margin across all cases when the mth variable is randomly permuted.**

Besides knowing which variables are important, RF also computes how the values of each variable effects the prediction.

PROXIMITY MEASURE

➔ These are one of the most useful tools in random forests.

➔ Since the trees are grown to maximum depth, the terminal nodes are small.

➔For each tree grown, pour all the data down the tree.

➔If two data points x_n and x_k occupy the same terminal node, increase prox(x_n x_k) by one.

➔At the end of forest growing, these proximities, form an intrinsic similarity measure between pairs of data vectors.

➔This is used to:

i) Estimate missing values.

ii) Give informative data views via metric scaling.

iii) Locate outliers.

OUTLIER LOCATION

Outliers are generally defined as cases that are removed from the main body of the data. Since the data in some classes is more spread out than others, outlyingness is defined only with respect to other data in the same class as the given case.

To define a measure of outlyingness,

i) Compute, for a case n, the sum of the squares of prox(n,k) for all k in the same class as case n.

ii) Take the inverse of this sum--it will be large if the proximities prox(n,k) from n to the other cases k in the same class are generally small. Denote this quantity by out(n).

iii) For all n in the same class, compute the median of the out(n), and then the mean absolute deviation from the median.

iv) Subtract the median from each out(n) and divide by the deviation to give a normalized measure of outlyingness. The values less than zero are set to zero.

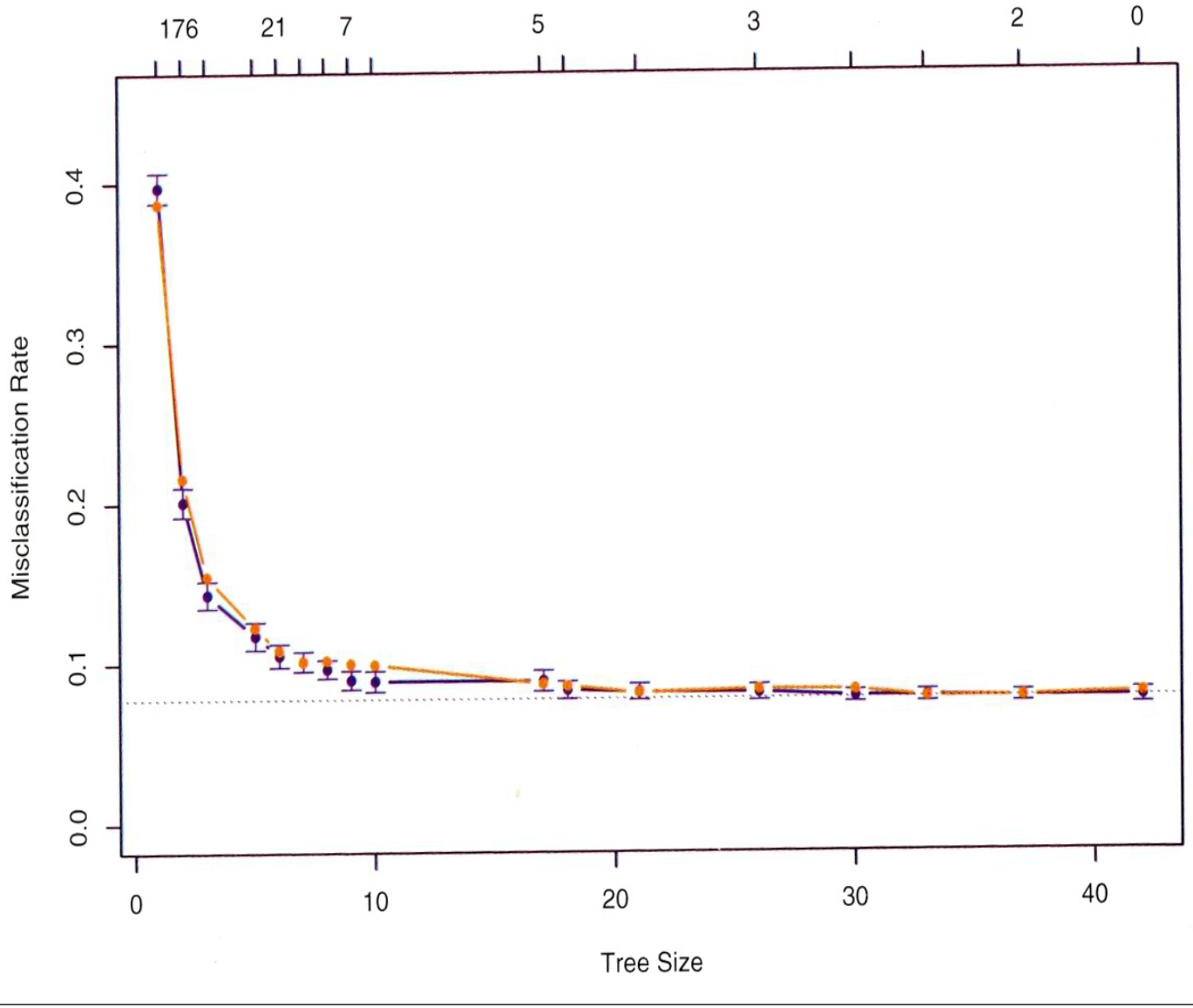
Generally, cases with value above 10 considered as outliers..

Missing value replacement

- Random forests has two ways of replacing **missing values.**
- In the first method, replace every missing value in the mth coordinate by the most frequent value(if it is categorical).
- In the second method, an iterative process is used
- If the mth coordinate in instance x_n is missing then it is estimated by a weighted average over the instances x_k with non-missing mth coordinate where the weight is prox(x_n, x_k).
- The replaced values are used in the next iteration of the forest which computes new proximities.
- The process it automatically stopped when no more improvement is possible or when five iterations are reached.

Classification Tree

Example of error rate Vs. tree size. α



Random Forest- Variance reduction

The average of random variables. B i.i.d. random variables, each with variance σ^2,

The mean has variance $\frac{1}{B}\sigma^2$

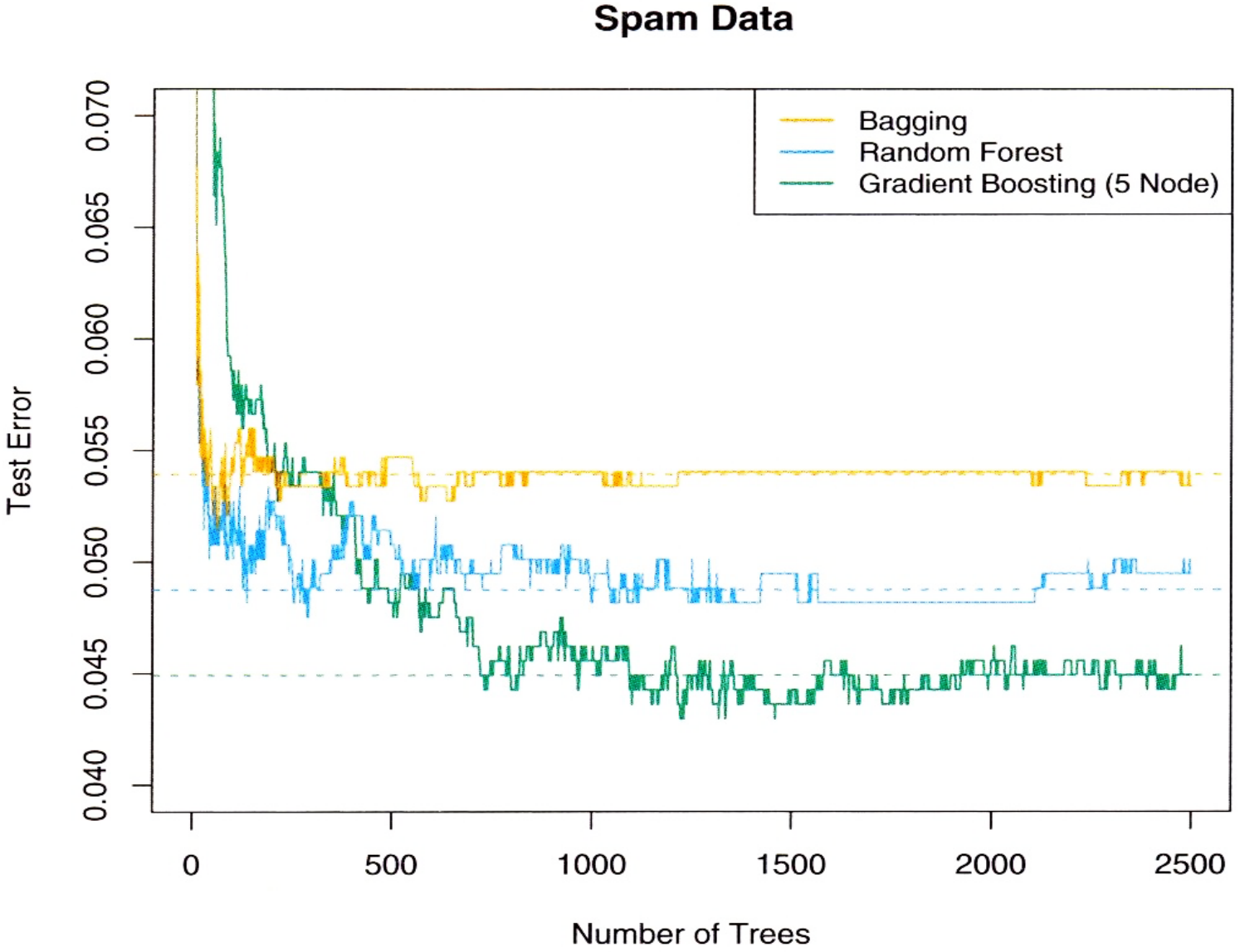
B i.d. random variables, each with variance σ^2, with pairwise correlation ρ

The mean has variance $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$

Random forest aims at reducing the correlation to reduce variance. This is achieved by random selection of variables.

Random Forest

Example comparing RF to boosted trees.



Reference

- Leo Breiman, *Random Forests*, Machine Learning, 45, 5-32, 2001

How Random Forest Works

Step 1: Build decision tree from a random subset of predictor variables.

Size of tree = sqrt (classification outcome) or 1/3 (continuous outcome) of the number of predictor variables.

Step 2: Use N random cases from the dataset, drawing with replacement. For each tree, approx. 1/3 of the dataset isn't used (Bootstrapping)

Step 3: Record the result of each unused case. After building the tree, 'run' the unused cases: record result.

Step 4: Repeat this process 500-1000 times

- Probabilities are generated by the total proportion of yes votes.
- Regression generated by average prediction.

Using random features

- Random split selection does better than bagging; introduction of random noise into the outputs also does better; but none of these do as well as Adaboost *by adaptive reweighting (arcing) of the training set*.
- To improve accuracy, the randomness injected has to minimize the correlation $\bar{\rho}$ while maintaining strength.
- The forests consists of *using randomly selected inputs* or *combinations inputs at each node* to grow each tree.

Variable importance

Because of the need to know which variables are important in the classification, RF calculates the variable importance.

➔To estimate the importance of the mth variable, in the oob cases for the kth tree, randomly permute all values of the mth variable

➔Put these altered oob x-values down the tree and get classifications.

➔Proceed as though computing a new internal error rate.

➔The amount by which this new error exceeds the original test set error is defined as the importance of the mth variable.

Besides knowing which variables are important, RF also computes how the values of each variable effects the prediction.

This is also computed under the hypothesis that the two variables are independent of each other and the latter subtracted from the former. A large positive number implies that a split on one variable inhibits a split on the other and conversely.

Bagging

"Bootstrap aggregation."

Resample the training dataset. Build a prediction model on each resampled dataset.

Average the prediction.

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{b}(x)$$

Bagging only differs from the original estimate when $\hat{f}(x)$ is a non-linear or adaptive function of the data!

Tree is a perfect candidate for bagging – each bootstrap tree will differ in structure.

Random Forest

Variable importance – find the most relevant predictors.

At every split of every tree, a variable contributed to the improvement of the impurity measure.

Accumulate the reduction of i(N) for every variable, we have a measure of relative importance of the variables.

The predictors that appears the most times at split points, and lead to the most reduction of impurity, are the ones that are important.

Missing value replacement

- Random forests has two ways of replacing missing values.
- If the mth variable is categorical, the replacement is the most frequent non-missing value in class j. These replacement values are called fills.
- The second way of replacing missing values is computationally more expensive but has given better performance than the first.
- If x(m,n) is a missing categorical variable, replace it by the most frequent non-missing value where frequency is weighted by proximity.
- Now iterate-construct a forest again using these newly filled in values, find new fills and iterate again.
- When there is a test set, there also exist two different methods of replacement depending on whether labels exist for the test set.