Alternatives to support vector machines in neuroimaging
ensembles of decision trees for classification and information mapping with predictive models

Jonas Richiardi
FINDlab / LabNIC
http://www.stanford.edu/~richiard
Decision trees deserve more attention

Scopus June 2013:

(mapping OR "brain decoding" OR "brain reading" OR classification OR MVPA OR "multi-voxel pattern analysis") AND (neuroimaging OR "brain imaging" OR fMRI OR MRI or "magnetic resonance imaging")

+ ("support vector machine" OR "SVM") = 657 docs
(1282 if adding EEG OR electroencephalography OR MEG OR magnetoencephalography)

+ ("random forest" OR "decision tree") = 71 docs
(199 if adding EEG OR electroencephalography OR MEG OR magnetoencephalography)

Roughly speaking, more used at MICCAI
(segmentation, geometry extraction, image reconstruction, skull stripping...) than at HBM
Tutorial agenda

Lecture
Basics
Growing trees
Ensembling
The random forest
Other forests
Tuning your forests
Information mapping
Correlated features

Practical
Datasets
Matlab/
PRoNTo
Python/Scikit
Information mapping relates model input to output

In supervised learning for classification, we seek a function mapping voxels to class labels

\[ f(x) : \mathbb{R}^D \rightarrow y \]

\[ y = \{0, 1\}, \quad x \in \mathbb{R}^D \quad S = \{(x_n, y_n)\}, \quad n = 1, \ldots, N \]

As neuroimagers, if some voxels in \( x \) contain information about \( y \), the function should reflect it, and we are interested in \textit{mapping} it

\[ f(x) = \text{sgn}(w^T x + b) \]

tells us about relative discriminative importance

[Mourao-Miranda et al., NeuroImage, 2005]
Mutual information measures uncertainty reduction

We can also explicitly measure the amount of information $x$ and $y$ share using mutual information.

$$I(Y; X) \equiv H(Y) - H(Y|X)$$

$$H(Y) \equiv \sum_{y} P(y)\log \frac{1}{P(y)}$$

high if classes are balanced (high uncertainty about class membership)

$$H(Y|X) \equiv \sum_{y,x} P(y, x)\log \frac{1}{P(y|x)}$$

average uncertainty remaining about class label if we know the voxel intensity

$$I(Y; X) \equiv \sum_{y,x} P(y, x)\log \frac{P(y, x)}{P(y)P(x)}$$

average reduction in uncertainty about class label if we know voxel intensity
Growing trees recursively reduces entropy.

Decision trees seek to partition voxel space by intensity values to decrease uncertainty about class label.

This leaves a few questions... How to measure goodness of splits? How to choose voxels and where to cut? When to stop growing?

> rpart::rpart
>> stats::classregtree
>>> sklearn::tree
Split goodness can be measured

**Entropy impurity:**  \( H(S) \equiv \sum_y P(y) \log \frac{1}{P(y)} \)

**Gini impurity:**  \( G(S) \equiv \sum_{y_i \neq y_j} P(y_i) P(y_j) \)

Information Gain (decrease in impurity):

\[
\Delta I(S; x, \tau) \equiv H(S) - \sum_{i=L,R} \frac{|S^i|}{|S|} H(S^i)
\]

[Hastie et al., 2001]
Information gain helps choose the best split

Before split

Split 1

Info Gain = 0.40

top

bottom

Split 2

Info Gain = 0.69

left

right

[Criminisi & Shotton, 2013]
Stopping and pruning matter

We can stop growing the tree

- When the info gain is small
- When the number of points in a leaf is small (relative or absolute) - dense regions of voxel space will be split more
- When (nested) CV error does not improve any more
  ...but stopping criteria are hard to set

We can grow fully and then prune

- Merge leaves where minimal impurity increase ensues

We can leave unpruned

These choices generally matter more than split goodness criterion (see CART vs C4.5)
Trees relate to other models

We can view trees as kernels: build a feature space mapping with indicator functions

\[ \Phi(x) = (1_1(x) \ldots 1_B(x))^T \]

Then \( k_b(x, x') = \Phi(x)^T \Phi(x') \) is a positive kernel (only = 1 if \( x \) and \( x' \) in same leaf). Can also do ‘soft’ version

We can also view trees as encoding conditional probability distributions, e.g. represented by Bayesian Networks:

\[
P(x_1, \ldots, x_D, y) = P(x_1) \ldots P(x_D)P(y|x_1, \ldots, x_D)
\]

Kernel view: [Geurts et al., 2006]

BN view: [Richiardi, 2007], others
Trees are rule-based classifiers

[Douglas et al. 2011]
Many trees are more complex

**Multivariate trees** query >1 voxels per node

Splits don’t have to be axis-parallel (can be oblique)

**Model trees** use MV regression in leaves

**Functional Trees** can use several voxels either at nodes or leaves

At each node, use \( \Delta I \) to split on either a voxel \( x \), or a logistic regression estimate of class probability \( P(y) \)

Multivariate nodes (FT-inner) reduce bias

Multivariate leaves (FT-leaves) reduces variance

Functional trees: [Gama, 2004]
## Single trees vs SVMs

<table>
<thead>
<tr>
<th></th>
<th>SVM</th>
<th>Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretability</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Irrelevant voxels</td>
<td>+(-)</td>
<td>+</td>
</tr>
<tr>
<td>Input scaling</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Speed</td>
<td>+ (linear)</td>
<td>+</td>
</tr>
<tr>
<td>Generalisation error</td>
<td>++</td>
<td>-</td>
</tr>
<tr>
<td>Information mapping</td>
<td>+(-)</td>
<td>-</td>
</tr>
</tbody>
</table>

[Kuncева & Rodríguez 2010]
Tutorial agenda

**Lecture**
- Basics
- Growing trees
- **Ensembling**
- The random forest
- Other forests
- Tuning your forests
- Information mapping
- Correlated features

**Practical**
- Datasets
- Matlab/PRoNTo
- Python/Scikit
Single trees tend to have high variance

The **bias-variance tradeoff** applies as usual:

We can decrease prediction error arbitrarily on a given dataset, thus yielding low **bias**.

However, this systematically comes at a price in **variance**: the parameters of $f$ can change a lot if the training set varies.

Single trees are not ‘stable’ - they tend to reduce bias by increasing variance

[Duda et al, 2001]
Ensembling exploits diversity

Train set of classifiers, combine predictions, get reduced ensemble variance and/or bias

Tree diversity has multiple sources

Training set variability/resampling, random projection, choice of cut point (, pruning strategy...)

18
Bagging classifiers generates diversity

Bagging = Bootstrap aggregating

1. Resample with replacement $B$ times from training dataset $S$, yielding $\{S_b\}, \ b = 1, \ldots, B$

2. Train $B$ base classifiers $\{f_b\}$

3. Get $B$ predictions $\hat{F} = \{\hat{f}_1(x), \ldots, \hat{f}_B(x)\}$

4. Combine by majority vote $\hat{f}(x) = Mo(\hat{F})$

If the base classifiers have high variance, accuracy tends to improve with bagging since this generates diversity

Good news for trees!

[Breiman, 1996]
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics</td>
<td>Datasets</td>
</tr>
<tr>
<td>Growing trees</td>
<td>Matlab/PRoNTo</td>
</tr>
<tr>
<td>Ensembling</td>
<td>Python/Scikit</td>
</tr>
<tr>
<td><strong>The random forest</strong></td>
<td></td>
</tr>
<tr>
<td>Other forests</td>
<td></td>
</tr>
<tr>
<td>Tuning your forests</td>
<td></td>
</tr>
<tr>
<td>Information mapping</td>
<td></td>
</tr>
<tr>
<td>Correlated features</td>
<td></td>
</tr>
</tbody>
</table>
The random forest bags trees

RF combines several diversity-producing methods

1. Generate B bootstrap replicates

2. At each node, randomly select a few voxels. Typically \( K = \lfloor \log_2 D + 1 \rfloor \) or \( \lfloor \sqrt{D} \rfloor \). Since \( K \ll D \), randomisation is high.

3. No pruning

With a ‘large enough’ number of trees, RFs typically performs well with no tuning on many datasets

Random projection: [Ho, 1998]

> adabag::bagging
>> stats::TreeBagger, PRoNTo::machine_RT_bin
>>> sklearn::RandomForestClassifier

Random projection: Kuncheva & Rodriguez 2010
RFs can be seen as probabilistic models

The leaf of each tree $b$ can be seen as a posterior (multinomial distribution) $p_b(y|x)$

Ensemble probability:

$$p(y|x) = \frac{1}{B} \sum_b p_b(y|x)$$

More trees = smoother posterior = less over-confidence
RF works for fMRI classification (I)

Data: event related fMRI, belief vs disbelief in statements, 14 subjects

Features: ICA timecourse value at button press

[Douglas et al. 2011]
RF works for multimodal classification

Data: 14 HC young, 13 AD old, 14 HC old. Visual stimulation + keypress. fMRI.

Features: fMRI GLM activation-related (n suprathreshold voxels, peak z-score,...), RT, demographics... + feature selection


[97-99% acc]

[Tripoliti et al. 2011]

activation features only + other features
RF really works for multimodal classification

Data: ADNI, 37 AD, 75 MCI, 35 HC. MRI, FDG-PET, CSF measures, 1 SNP

Features: RF as kernel + MDS

[Gray et al 2013]
Tutorial agenda

Lecture

Basics
Growing trees
Ensembling
The random forest

Other forests
Tuning your forests
Information mapping
Correlated features

Practical
Datasets
Matlab/PRoNTo
Python/Scikit
Extremely Randomised Trees increase diversity

Tree variance is due in large part to cutpoint choice. We can generate even more diversity with Extra-trees

Select both K voxels and cutpoints at random, pick best*.

Stop growing when leaves are small

When K=1, called totally randomized trees

+ Accuracy and variance reduction competitive with and sometimes better than RF, faster than RF

Extra-trees: [Geurts et al., 2006]
*Dietterich 1998 - the opposite: select top-K best splits, then pick at random

extraTrees::extraTrees
>>PRoNTo::machine_RT_bin
>>>sklearn::ExtraTreesClassifier
Rotation forests do subspace PCA

We can also generate random rotations of the data to add diversity. For each tree:

1. Project training data $\mathbf{X}$ into $M$ random non-overlapping subspaces, each of size $K$
2. For each subspace: choose a subset of classes, draw 75% bootstrap, do PCA
3. Rearrange PCs into a block-diag matrix $\mathbf{R}$ and project whole training set to $\mathbf{XR}$.
4. Train the tree

\[\text{Weka via rJava} \quad \text{>>Weka via writeARFF or Java} \quad \text{>>>?}\]
RotFor works for fMRI

Data: Haxby (8 classes, 90 points per class, 43K voxels)

Tests: feature selection, ensembles vs SVM

RotFor

[Kuncheva&Rodriguez 2010]
Ensemble of FTs may improve accuracy

PCA-derived features are multivariate in the original space, so in fact RotFor does axis-parallel (univariate) cuts on MV data...

We can also try slightly more stable MV trees instead of univariate trees (trade diversity for accuracy)

On 62 UCI low-dimensional datasets, it seems that Bagging + FT-leaves works about the same as RotFor + univariate tree*. All other ensembles of univariate trees perform worse...

On high-dimensional fMRI connectivity data**, and low-dimensional graph/vertex attribute representations of fMRI connectivity***, bags of FTs work quite well

## Trees ensembles vs SVMs

<table>
<thead>
<tr>
<th></th>
<th>SVM</th>
<th>Tree ensembles</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Interpretability</strong></td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Irrelevant voxels</strong></td>
<td>-/+</td>
<td>+</td>
</tr>
<tr>
<td><strong>Input scaling</strong></td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td><strong>Speed</strong></td>
<td>+ (linear)</td>
<td>+ (parallel)</td>
</tr>
<tr>
<td><strong>Generalisation error</strong></td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td><strong>Information Mapping</strong></td>
<td>+/-</td>
<td>++ (see later)</td>
</tr>
</tbody>
</table>
## Tutorial agenda

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics</td>
<td>Datasets</td>
</tr>
<tr>
<td>Growing trees</td>
<td>Matlab/PRoNTo</td>
</tr>
<tr>
<td>Ensembling</td>
<td>Python/Scikit</td>
</tr>
<tr>
<td>The random forest</td>
<td></td>
</tr>
<tr>
<td>Other forests</td>
<td></td>
</tr>
<tr>
<td><strong>Tuning your forests</strong></td>
<td></td>
</tr>
<tr>
<td>Information mapping</td>
<td></td>
</tr>
<tr>
<td>Correlated features</td>
<td></td>
</tr>
</tbody>
</table>
For many tree ensembles, more trees (L) lead to more decrease in variance

Typically use several hundreds to reach plateau (Langs: 40K)

Large L “better approximates infinity” than small L

For RF, the out-of-bag error estimate’s bias decreases a lot with increasing trees - bootstrapping uses ~2/3 of data for each tree, more trees leads to better OOB estimate

This also gives a much smoother posterior distribution

For multivariate trees, use fewer trees

10-30 works well empirically on very different datasets
The optimal projection dimension, $K$, depends on the presence of irrelevant voxels. Many useful voxels - use small $K$. Information concentrated in few voxels - use large $K$. See e.g. [Geurts et al. 2006] for details.

$$D = 2 - 10K, \ N = 40 - 100, \ C = 2 - 8$$
Tutorial agenda

Lecture
Basics
Growing trees
Ensembling
The random forest
Other forests
Tuning your forests
Information mapping
Correlated features

Practical
Datasets
Matlab/
PRoNTo
Python/Scikit
Tree ensembles directly provide information maps

The split criterion and related measures are natural indicators of the ‘usefulness’ of voxels in the discrimination task

But they are unsigned

They are computed at each node of the tree, so we can aggregate them over trees to get stable estimates

Different ensembles provide different information maps, and we can use other data than split criteria to map
GI of a voxel: infoGain (compute with Gini impurity) for this voxel, averaged over all trees in ensemble.
Information mapping: RF/GI/var

Weights map

Anatomical img

Gini importance vs voxel index
Information Mapping: $L_2$ SVM
Data: ADNI, 37 AD, 75 MCI, 35 HC. MRI, FDG-PET, CSF measures, 1 SNP

[Gray et al 2013]
VI of a voxel*: average loss of accuracy on OOB samples when randomly permuting values of the voxel

This is suboptimal with correlated voxels

  Permuting one single variable ignores correlations

  With several relevant & correlated voxels, they could be deweighted because removing one does not deteriorate accuracy

  VI is well-correlated with GI**

More on this later

*[Breiman, 2001]  **[Strobl et al., 2007]
Leaves in an FT can be regression models

These can be trained using any method, in practice LogitBoost (iterative reweighting) works well

The importance of a voxel is its average regression weights across trees and folds

[Richiardi et al. 2011a]
Finally we could also map results directly from classifier with best accuracy

Here: Haxby data, SVM-RFE 200, RF1000, intersection of selected features across 10 folds, one slice

[Kuncheva et al. 2010]
# Tutorial agenda

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics</td>
<td>Datasets</td>
</tr>
<tr>
<td>Growing trees</td>
<td>Matlab/</td>
</tr>
<tr>
<td>Ensembling</td>
<td>PRoNTo</td>
</tr>
<tr>
<td>The random forest</td>
<td>Python/Scikit</td>
</tr>
<tr>
<td>Other forests</td>
<td></td>
</tr>
<tr>
<td>Tuning your forests</td>
<td></td>
</tr>
<tr>
<td>Information mapping</td>
<td></td>
</tr>
<tr>
<td><strong>Correlated features</strong></td>
<td></td>
</tr>
</tbody>
</table>
Correlated features are picked up by tree ensembles

With regularisers in SVMs, correlated features will be deweighted ($L_2$) or left out ($L_1$)

Tree ensembles have grouping effect*, where correlated but informative features can survive with high weight

Empirically this seems to depend on tree depth...

*Langs et al., 2011 [Pereira & Botvinick, 2011]
There are ways of dealing with correlated features

Several proposals from bioinformatics have attempted to tackle the GI/VI measure bias

Conditional Variable Importance only permutes a correlated variable within observations where its correlands have a certain value (accounts for correlation structure)

Permutation IMPortance fixes for under-importance of grouped vars by permuting class labels, then constructing a null distribution of GI values

These methods can be used in neuroimaging directly...

Cond. Var. Imp. [Strobl et al, 2008]
PIMP [Altmann et al., 2010]
Tutorial agenda

Lecture
Basics
Growing trees
Ensembling
The random forest
Other forests
Tuning your forests
Information mapping
Correlated features

Practical
Datasets
Matlab/PRoNTo
Python/Scikit
## Tutorial agenda

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Practical Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics</td>
<td>Matlab/PRoNTo</td>
</tr>
<tr>
<td>Growing trees</td>
<td>Python/Scikit</td>
</tr>
<tr>
<td>Ensembling</td>
<td></td>
</tr>
<tr>
<td>The random forest</td>
<td></td>
</tr>
<tr>
<td>Other forests</td>
<td></td>
</tr>
<tr>
<td>Tuning your forests</td>
<td></td>
</tr>
<tr>
<td>Information mapping</td>
<td></td>
</tr>
<tr>
<td>Correlated features</td>
<td></td>
</tr>
</tbody>
</table>
Datasets

1. SINGLE SUBJECT: SPM Auditory - “Mother of All Experiments” - 1 subject, 2T scanner, TR=7s, 6 blocks of 42 s rest, 42s auditory stimulation.
Task: two-class intra-subject decoding: auditory vs rest.

2. GROUP COMPARISON: Buckner checkerboard - 41 subjects from three groups, young (18-24), elderly healthy (66-89) and elderly demented (age 68-83). Four runs per subject, 128 volumes per run with TR=2.68s.
Task: classify young (n=28) versus old (n=30) group based on ‘first level’ beta maps

Original auditory data at www.fil.ion.ucl.ac.uk/spm/data/auditory/
Original visual data at fmridc.org

[Buckner et al., 2001]
Tutorial agenda

Lecture
Basics
Growing trees
Ensembling
The random forest
Other forests
Tuning your forests
Information mapping
Correlated features

Practical
Datasets
Matlab/PRoNTo
Python/Scikit
The PRoNTo toolbox

Matlab code, open source, GUI / batch / scripting

Users can quickly test machine learning methods without coding

http://www.mlnl.cs.ucl.ac.uk/pronto/
1. Make sure your path is setup properly
   
   >> which spm
   >> which pronto

2. Start PRoNTo
   
   >> pronto
<table>
<thead>
<tr>
<th>Lecture</th>
<th>Practical</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics</td>
<td>Datasets</td>
</tr>
<tr>
<td>Growing trees</td>
<td>Matlab/PRoNTo</td>
</tr>
<tr>
<td>Ensembling</td>
<td>Python/Scikit</td>
</tr>
<tr>
<td>The random forest</td>
<td></td>
</tr>
<tr>
<td>Other forests</td>
<td></td>
</tr>
<tr>
<td>Tuning your forests</td>
<td></td>
</tr>
<tr>
<td>Information mapping</td>
<td></td>
</tr>
<tr>
<td>Correlated features</td>
<td></td>
</tr>
</tbody>
</table>

Datasets
Matlab/PRoNTo
Python/Scikit
A very good alternative for python fans is to use Scikit-learn + niPy. It has RF, Extra-trees, and others.

For people that missed Gaël’s tutorial at PRNI 2011: [http://nisl.github.io/](http://nisl.github.io/)

PyMVPA also has access to Extra-trees and RF.
Conclusions

Tree ensembles can offer competitive decoding performance with SVMs, and are good for multimodal classification.

They produce information maps which are typically sparser than $L_2$ SVMs (is this good or bad?), and can have different interpretation.

Implementations abound in the language of your choice, including R, Matlab, Python.

So... take a walk in the forest for your next project.
Thanks

FINDDlab, Stanford University
  A. Altmann
Computational Image Analysis and Radiology, Medical U. of Vienna / CSAIL, MIT
  G. Langs
Montefiore Institute, U. of Liège
  P. Geurts
FIL, UCL
  G. Rees
PRoNTo team members @ UCL, KCL, U. Liège, NIH
Computer Vision Group, U. Freiburg
  A. Abdulkadir

Modelling and Inference on Brain networks for Diagnosis, MC IOF #299500

Hastie et al. (2011). The Elements of Statistical Learning, Springer.

Altmann, A. et al. (2010). Permutation importance: a corrected feature importance measure
Diaz-Uriarte et al. (2006). Gene selection and classification of microarray data using random
forest. BMC Bioinformatics 7:3
Douglas, P.K. et al. (2011). Performance comparison of machine learning algorithms and number of
independent components used in fMRI decoding of belief vs. disbelief. NeuroImage 56.
Gray, K. et al. (2013). Random forest-based similarity measures for multi-modal classification of
Magnetic Resonance Imaging 28
NeuroImage 56(2)
Mourao-Miranda, J. et al. (2005). Classifying brain states and determining the discriminating
NeuroImage 56(2).
28(10)
Tripoliti, E. E. et al. (2011). A supervised method to assist the diagnosis and monitor progression of
Alzheimer’s disease using data from an fMRI experiment. Artificial Intelligence in Medicine 53