Modern Machine Learning: Partition & Vote

Carey E. Priebe
Johns Hopkins University
and
Joshua T. Vogelstein
Johns Hopkins University
and
Florian Engert
Harvard University
and
Christopher M. White
Microsoft Research

April 27, 2020

Abstract

We present modern machine learning, focusing on the state-of-the-art classification methods of deep networks and decision forests, as partition and vote schemes. This illustrative presentation allows for both a unified basic understanding of how these methods work from the perspective of classical statistical pattern recognition as well as useful basic insight into their relationship with each other ... and with brain functioning.

Keywords: Deep Networks; Decision Forests; Classification, Brain functioning.

1 Introduction

The classical statistical formulation of the classification problem consists of

$$(X, Y), (X_1, Y_1), \cdots, (X_n, Y_n) \overset{iid}{\sim} F_{XY},$$

where $\mathcal{T}_n = \{(X_i, Y_i)\}_{i \in \{1, \cdots, n\}}$ is the training data and (X, Y) represents the to-be-classified test observation X with true-but-unobserved class label Y. We consider the simplest setting in which X is a feature vector in $\Re^d$ and Y is a class label in $\{0, 1\}$. The goal is to learn a classification rule $g_n = g(\cdot; \mathcal{T}_n)$ mapping feature vectors to class labels such that the probability of misclassification $L(g_n) = P[g(X; \mathcal{T}_n) \neq Y | \mathcal{T}_n]$ is small.

Stone's Theorem for universally consistent classification (Stone, 1977; Devroye et al., 1997) demonstrates, loosely speaking, that a successful classifier can be constructed by first partitioning the input space into cells – the partition depends on n – such that the number of training data points in each cell goes to infinity but slowly in n, and then estimating the posterior $\eta(x) = P[Y = 1 | X = x]$ locally by voting based on the training class labels associated with the training feature vectors in cell $C(x) \subset \Re^d$ in which the test observation falls. That is, letting $S(x) = \{X_1, \cdots, X_n\} \cap C(x)$ be the subset of X_i's falling in cell $C(x)$, and letting $N(x) = |S(x)| = \sum_i I\{X_i \in C(x)\}$ be the cardinality of this set, we consider the posterior estimate $\widehat{\eta}(x) = (1/N(x)) \sum_{i: X_i \in C(x)} I\{Y_i = 1\}$. Then, under some technical conditions on the manner of choosing the sequence of partitions $\mathcal{P}_n = \{C_{n,1}, \cdots, C_{n,K_n}\}$, the plug-in rule $g(X; \mathcal{T}_n) = I\{\widehat{\eta}(X) > 1/2\}$ is universally consistent: $L(g_n) \to L^*$ almost surely for any F_{XY}, where L^* is the Bayes optimal probability of misclassification.

2 Decision Forests & Deep Networks

In the context of our formulation of the classification problem, we provide a unified description of the two dominant methods in modern machine learning, the decision forest (DF) and the deep network (DN), as ensemble partition and vote schemes. This allows for useful basic insight into their relationship with each other ... and with brain functioning.

2.1 Decision Forests

Decision forests, including random forests and gradient boosting trees, demonstrate state-of-the-art performance in a variety of machine learning settings. DFs have typically been implemented as ensembles of axis-aligned decision trees – trees that split along canonical feature dimensions only – but modern extensions employ axis-oblique splits (Breiman, 2001; Criminisi and Shotton, 2013; Athey et al., 2019; Tomita et al., 2020).

Given the training data $\mathcal{T}_n$, each tree in a random forest constructs a partition $\mathcal{P}_{n,t}$ by successively splitting the input space based on a random subset of the data (per tree) and then choosing a hyperplane split based on a random subset of the dimensions (per node) using some criterion for split utility at each node such as maximal reduction in class confusion for the training data falling to that node. The details provide ample fodder for myriad implementations, and a simple depiction necessarily suffers from an inability to visualize input space $\Re^d$ for $d > 2$, but Figure 1 illustrates the idea.

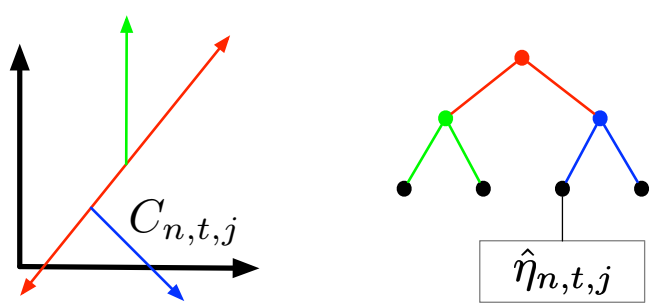


Figure 1: A tree in the forest. Given the random subset $\mathcal{T}_{n,t}$ of the training data allocated to tree t, the root node (red) performs a hyperplane split of $\Re^d$ based on a random subset of dimensions; the two daughter nodes (green and blue) split their respective partition cells based on a separate random subset of dimensions allocated to each node; etc. In the end, this tree results in a partition of $\Re^d$ with the leaf nodes corresponding to partition cells for which training data class labels yield local posterior estimates. The forest classifies the test observation feature vector X by voting over trees using the cells $C_{n,t}(X)$ in which X falls.

The depth of each tree is a function of n and involves a tradeoff between leaf purity and regularization to alleviate overfitting. Details aside, each tree results in a partition $\mathcal{P}_{n,t}$ of $\Re^d$ and each partition cell – each leaf of each tree – admits a posterior estimate $\widehat{\eta}_{n,t,j} = (1/N_{n,t,j}) \sum_{i:X_i \in C_{n,t,j}} I\{Y_i = 1\}$ based on the class labels of the training data feature vectors that fall into cell $C_{n,t,j}$.

The conceit of the forest is that of resampling: by choosing a separate subset $\mathcal{T}_{n,t}$ of the training data for each tree t, the ensemble forest posterior estimate is superior to that of any individual tree. Regardless, the overall classifier is seen to be a partition and vote scheme, and under appropriate conditions on the manner of choosing the partitions $\mathcal{P}_{n,t}$ we have $L(g_n^{DF}) \to L^*$.

2.2 Deep Networks

Deep networks are extraordinarily popular and successful in modern machine learning. (LeCun et al., 2015; Sze et al., 2017; Montúfar et al., 2014; Montúfar, 2017). As depicted in Figure 2, each internal node $v_{\ell,k}$ in layer ℓ of the network gathers inputs $\tilde{x}_{\ell-1,j}, j = 1, \cdots, n_{\ell-1}$ from the previous layer, weighted by $w_{\ell-1,j,k}, j = 1, \cdots, n_{\ell-1}$, and outputs

$$\tilde{x}_{\ell,k} = f_{\ell,k}(\sum_j \tilde{x}_{\ell-1,j} w_{\ell-1,j,k}) = f_{\ell,k}(\tilde{X}_{\ell-1}^T W_{\ell-1,k}) = f(\tilde{X}_{\ell-1}^T W_{\ell-1,k} + b_{\ell,k}).$$

When using the so called ReLU (rectified linear unit) function $\max(0, \cdot) = (\cdot)^+$ as the activation function f, each node performs a hyperplane split based on a linear combination of its inputs, passing a non-zero value forward if the input is on the preferred side of the hyperplane; data in the cell defined by the collection of polytopes induced by nodes $\{v_{\ell-1,j}\}$, weighted, falls into node $v_{\ell,k}$ and is output based on a partition refinement. Letting $\tilde{X}_\ell = [\tilde{x}_{\ell,1}, \cdots, \tilde{x}_{\ell,n_\ell}]^T$ be the output of layer ℓ and $W_{\ell,j} = [w_{\ell,1,j}, \cdots, w_{\ell,n_\ell,j}]^T$ be the weights from layer ℓ to the jth node in layer $\ell + 1$, node $v_{\ell+1,j}$ sees $\tilde{X}_\ell^T W_{\ell,j}$ and outputs $\tilde{x}_{\ell+1,j} = (\tilde{X}_\ell^T W_{\ell,j} + b_{\ell+1,j})^+$. Thus a node in the last internal layer represents a network partition cell defined via composition by all nodes earlier in the network.

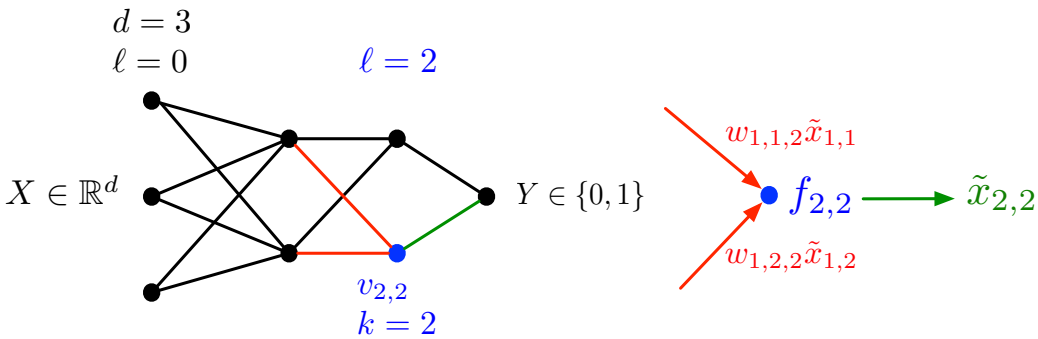


Figure 2: A (shallow) deep network. Given the training data $\mathcal{T}_n$, the $X_i \in \Re^d$ are passed through the network. At layer ℓ and node $v_{\ell,k}$ (the blue node is $v_{2,2}$) the inputs $w_{\ell-1,j,k}\tilde{x}_{\ell-1,j}$ (red) are transformed via hyperplane activation function $f_{\ell,k}$ and output as $\tilde{x}_{\ell,k}$ (green). Thus node $v_{2,2}$ receives non-zero input $w_{1,j,2}\tilde{x}_{1,j} = w_{1,j,2}(X^T W_{0,j} + b_{1,j})^+$ from node $v_{1,j}$ if and only if the linear combination of the multivariate X, $X^T W_{0,j}$, is on the preferred side of hyperplane defined by $f_{1,j}$ (and weight $w_{1,j,2}$ is non-zero). The output of node $v_{2,2}$ is $\tilde{x}_{2,2} = \left(\begin{bmatrix} (X^T W_{0,1} + b_{1,1})^+ \\ (X^T W_{0,2} + b_{1,2})^+ \end{bmatrix}^T W_{1,2} + b_{2,2} \right)^+$; that is, $v_{2,2}$ provides a further hyperplane refinement of $\Re^d$.

For conceptual unification with DFs, consider passing all the training data through the network; the collection of X_i's falling into these final network partition cells, together with their class labels, induces local posterior estimates for each of these cells. As with a single tree in a DF wherein the test observation feature vector X, suitably transformed, falls into one and only one leaf partition cell, Montúfar et al. (2014) argues that DNs with ReLu activation functions fold space such that a subset of $\Re^d$ maps to exactly one fold. Thus for a DN the input X falls into a final network partition cell in the last internal layer. Thus while for a DF the ensemble is realized by voting over a forest of trees, for a DN we have membership in a single final network partition cell. On the other hand, each cell in a DF is constructed via a simple local splitting process, while for the DN a complex parameter estimation solution is employed to tailor the individual final network partition cells to the training data.

The parameters for this partition and vote scheme are estimated ("learned") so as to make the network input/output fit the training data $\mathcal{T}_n$, with much current effort involving aspects of regularization to alleviate overfitting and capacity as a function of the number of layers and nodes per layer.

3 An Illustrative Comparative Example

Practical application of both DFs and DNs involve a multitude of design and computational considerations; Figure 3 presents a simple illustration of comparative performance. The experimental setup is as follows (see Park (2020) for full reproduction capability). We choose two CIFAR-10 classes (Krizhevsky, 2012, 2015). There are 6000 images per class, which are split into training (5000 per class) and testing (1000 per class). For DF we use the R package "ranger" (Wright and Ziegler, 2017); for DN we use the R package "keras" (Chollet et al., 2015). The figure shows classification performance $L(g)$ against the number of training samples n.

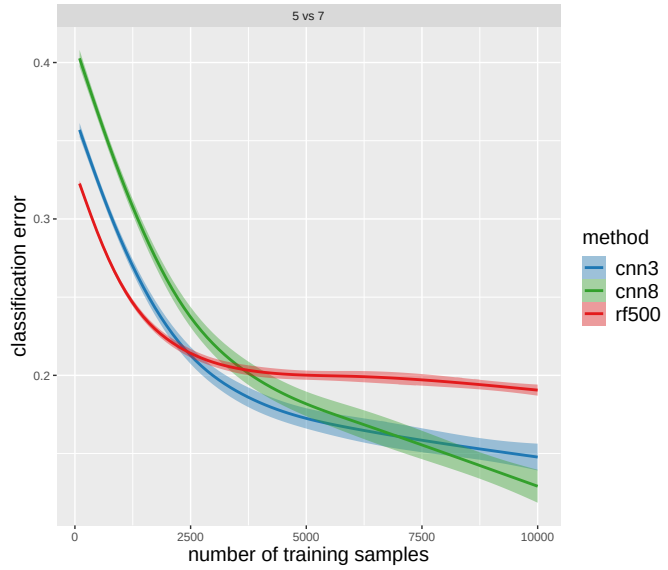


Figure 3: Performance comparison of a DF to two DNs: DF in red; DN in green and blue, with blue the simpler architecture and green the more complex. The x-axis is training set sample size n; the y-axis is classification error $L(g)$.

DFs, with the number of trees and their depths going to infinity as $n \to \infty$, are inherently nonparametric. DNs, with a fixed architecture, are parametric, albeit with a large enough number of parameters, and a mysterious enough implicit regularization effect, to blur this distinction in many settings. The simple comparative example presented in Figure 3 demonstrates the expected DN bias-variance tradeoff: a simpler DN outperforms a more complex DN when n is small, with this relationship switched for n large enough. DF (properly designed and relaxed in n) will eventually outperform any fixed-architecture DN as n increases; in this example the maximum n considered is too small for this limiting effect to obtain. For n very small, DF superiority is due to the fact that the variance of the DN that is required to get good performance for larger n renders its risk worse at low n.

4 A relationship with brain functioning

Learning in both DNs and DFs can be thought of as ensemble partition and vote functions implemented by a network of nodes. Intriguingly, learning in biological brains can be viewed similarly. In brains, a 'node' can correspond to a unit across many scales, ranging from synaptic channels (which can be selectively activated or deactivated due to the synapses' local history), to cellular compartments, individual cells, or cellular ensembles (Vogelstein et al., 2019). At each scale, brains learn by partitioning feature space. The feature space that is partitioned is the set of all possible sensory inputs; a 'part' corresponds to a subset of 'nodes' that tend to respond to any given input. An example is the selective response properties that define cortical columns–columns in the sensory cortex (Mountcastle, 1997). Brains also vote, where voting is a pattern of responses based on neural activation that indicate which stimulus evoked a response (Machens et al., 2005).

Figure 4 illustrates neural selectivity to features of sensory input in a larval zebrafish, and indicates a learned partition and vote framework for brain functioning (Naumann et al., 2016). Specifically, this image shows all motion-sensitive nodes ($n = 76,604$) in a brain. Each node is a dot, color coded for preferred motion direction. The image indicates that each node's activity indeed corresponds to a partition of feature space, which is actively refined through sensory experience and learning. A relationship with DFs and DNs, at

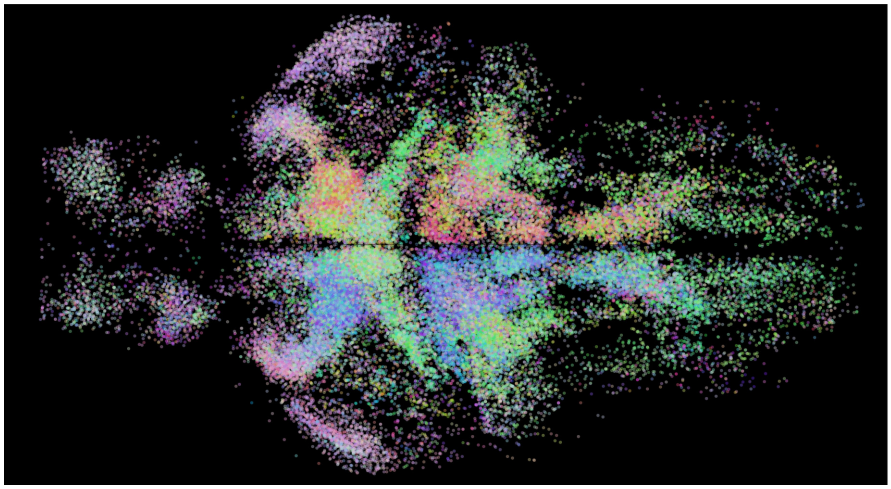


Figure 4: Whole-brain activity maps reveal processing stages underlying the optomotor response.

least at a basic level, is clear. The utility of this relationship, for either machine learning or neuroscience, is the subject of endless conjecture and refutation.

5 Conclusion

"Many important classification rules partition $\Re^d$ into disjoint cells $A_1, A_2, \ldots$ and classify in each cell according to the majority vote among the labels of the X_i's falling in the same cell" (Devroye et al. (1997), p. 94). The purpose of this short note is to describe state-of-the-art machine learning methods simplified in classical statistical pattern recognition terms. The depiction of deep networks and decision forests as partition and vote schemes allows for both a unified basic understanding of how these methods work from the perspective of classical classification as well as useful basic insight into their relationship with each other ... and with brain functioning. Elaborate generalization of the partition and vote framework is the subject of intense research – theoretical, methodological, and for myriad applications – but the fundamental idea is here presented in its most basic form.

References

Athey, S., J. Tibshirani, and S. Wager (2019). Generalized random forests. *Annals of Statistics 47*(2), 1148–1178.

Breiman, L. (2001). Random forests. *Machine Learning 45*(1), 5–32.

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`, `https://keras.io`.

Criminisi, A. and J. Shotton (2013). *Decision Forests for Computer Vision and Medical Image Analysis*. Springer.

Devroye, L., L. Györfi, and G. Lugosi (1997). *A Probabilistic Theory of Pattern Recognition*. Springer.

Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. *University of Toronto*.

Krizhevsky, A. (2015). CIFAR datasets. `https://www.cs.toronto.edu/~kriz/cifar.html`.

LeCun, Y., Y. Bengio, and G. Hinton (2015). Deep learning. *Nature 521*(7553), 436–444.

Machens, C. K., R. Romo, and C. D. Brody (2005). Flexible control of mutual inhibition: a neural model of two-interval discrimination. *Science 307*(5712), 1121–1124.

Montúfar, G. (2017). Notes on the number of linear regions of deep neural networks. In *SampTA 2017*, Tallinn, Estonia.

Montúfar, G., R. Pascanu, K. Cho, and Y. Bengio (2014). On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, pp. 2924–2932.

Mountcastle, V. (1997). The columnar organization of the neocortex. *Brain 120*(4), 701–722.

Naumann, E. A., J. E. Fitzgerald, T. W. Dunn, J. Rihel, H. Sompolinsky, and F. Engert (2016). From Whole-Brain Data to Functional Circuit Models: The Zebrafish Optomotor Response. *Cell 167*(4), 947–960.e20.

Park, Y. (2020). Random Forest & Convolutional Neural Network Demo. `http://www.cis.jhu.edu/~parky/RF/tasMML.html`.

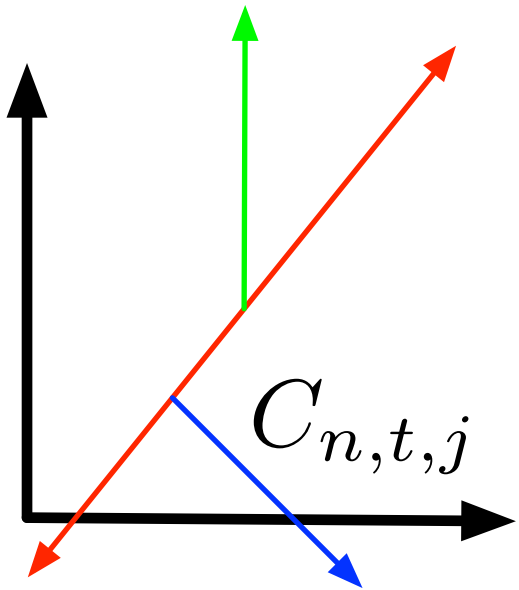
Stone, C. (1977). Consistent nonparametric regression. *Annals of Statistics 5*(2), 595–645.

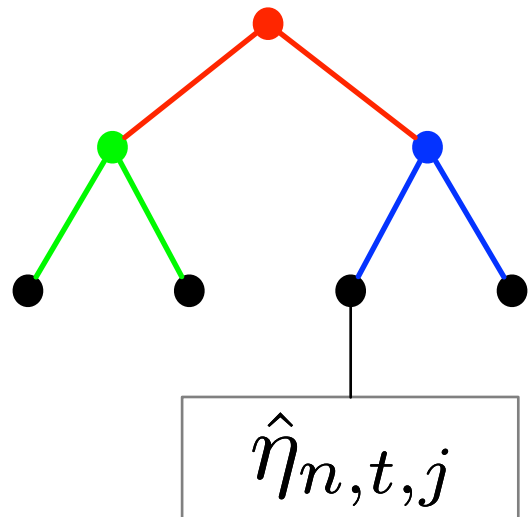
Sze, V., Y.-H. Chen, T.-J. Yang, and J. S. Emer (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE 105*(12), 2295–2329.

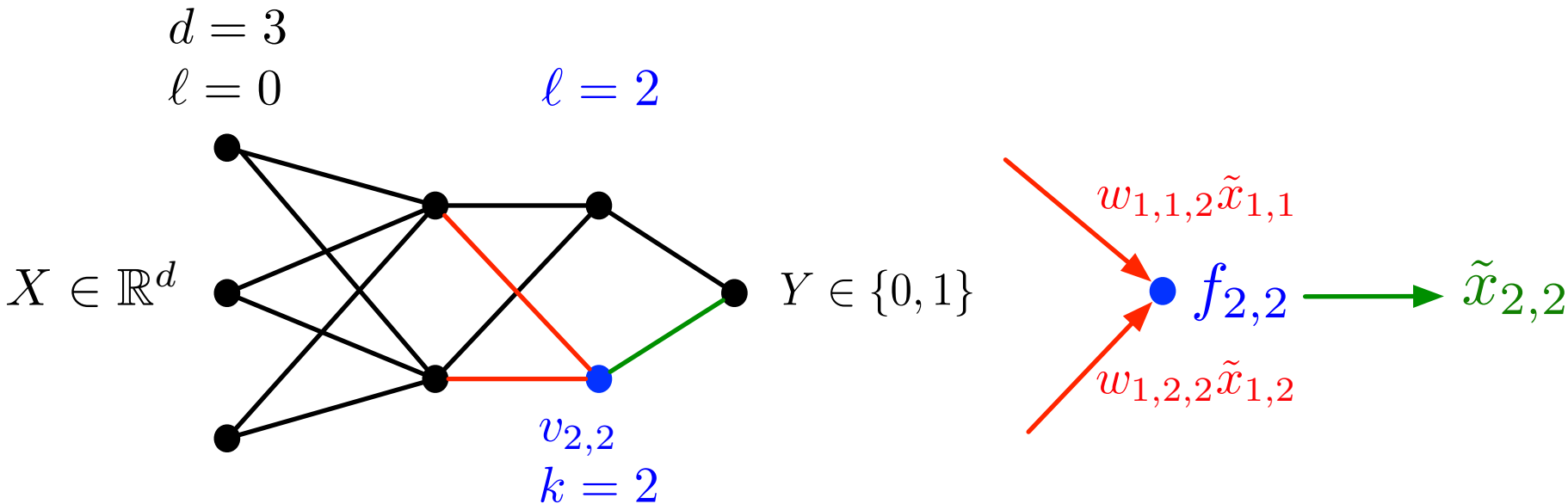
Tomita, T. M., J. Browne, C. Shen, J. Chung, J. L. Patsolic, B. Falk, J. Yim, C. E. Priebe, R. Burns, M. Maggioni, and J. T. Vogelstein (2020). Sparse projection oblique randomer forests. *Journal of Machine Learning Research to appear.*

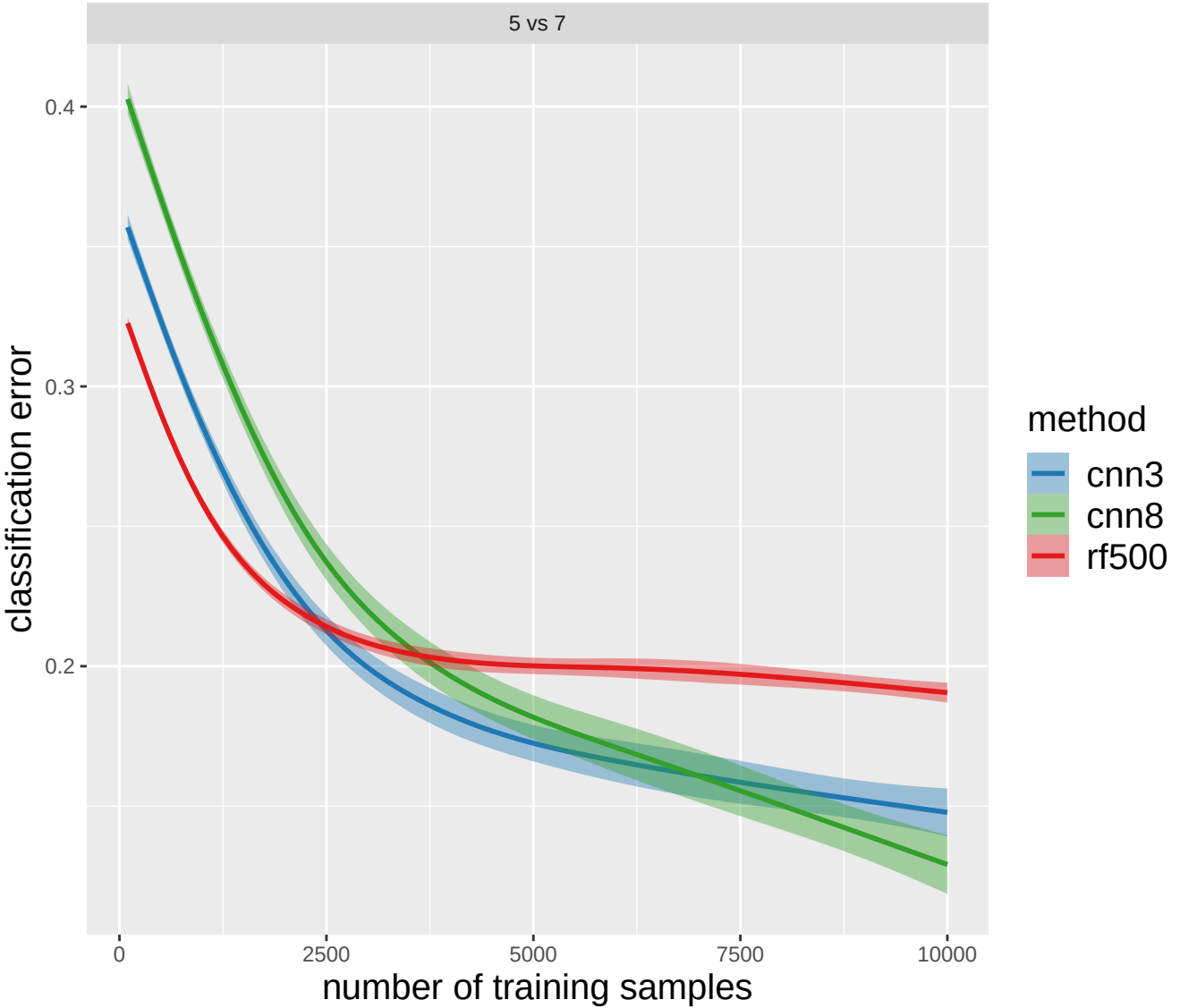
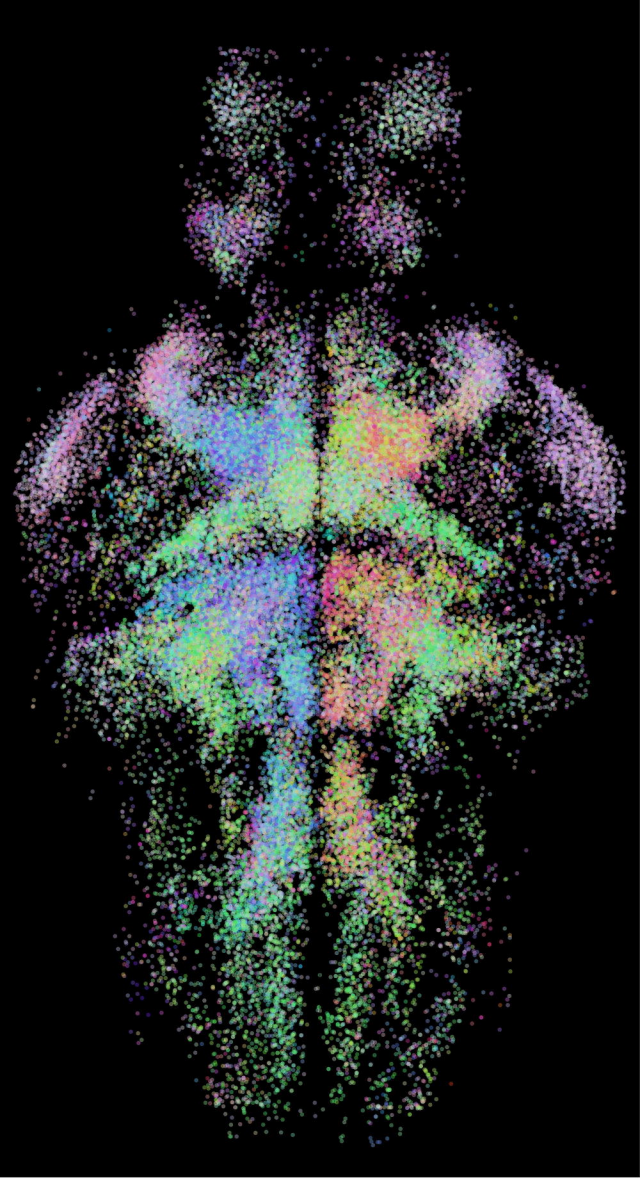
Vogelstein, J. T., E. W. Bridgeford, B. D. Pedigo, J. Chung, K. Levin, B. Mensh, and C. E. Priebe (2019). Connectal coding: discovering the structures linking cognitive phenotypes to individual histories. *Curr. Opin. Neurobiol. 55*, 199–212.

Wright, M. and A. Ziegler (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software 77*(1), 1–17.


$C_{n,t,j}$


$\hat{\eta}_{n,t,j}$


$d = 3$
$\ell = 0$
$\ell = 2$
$X \in \mathbb{R}^d$
$Y \in \{0, 1\}$
$v_{2,2}$
$k = 2$
$w_{1,1,2}\tilde{x}_{1,1}$
$f_{2,2}$
$\tilde{x}_{2,2}$
$w_{1,2,2}\tilde{x}_{1,2}$


5 vs 7
0.4
0.3
0.2
classification error
0
2500
5000
7500
10000
number of training samples
method
cnn3
cnn8
rf500