An Interactive Tutorial for Teaching Decision Trees and Random Forests

Wesley Beckner
Chemical Engineering,
University of Washington
wab665@uw.edu

Matt Murbach
Chemical Engineering,
University of Washington
mmurbach@uw.edu

Janet Matsen
Chemical Engineering,
University of Washington
jmatsen@uw.edu

ABSTRACT
This interactive visualiation teaches how machine learning can be used to predict photoluminescence. All the code can be accessed at `https://github.com/CSE512-16S/fp-wesleybeckner-mdmurbach-janetmatsen`, and the live visualization is at `http://cse512-16s.github.io/fp-wesleybeckner-mdmurbach-janetmatsen/`.

Author Keywords
machine learning; decision trees; random forests; D3; education; photoluminescence

INTRODUCTION

Machine learning and high performance computing revolutionize strategies to engineer new materials. By creating automated and adaptive experimental architectures, previously insurmountable design spaces can now be considered [3]. In this spirit, we created a tutorial to illustrate the performance of random forests as implemented in sklearn.ensemble.RandomForestRegressor [4] at predicting photoluminescence for perovskite solar cells. The tutorial is meant to be a creative exploration space to introduce other researchers and students to one of the most commonly used machine learning algorithms.

RELATED WORK

Excitement about machine learning has led to many resources aimed at educating researchers about the underlying methods. Interactive designs allow additional opportunity to communicate the structure of algorithms and implications of hyperparameters.

Two compelling examples are seen on the websites Visual Introduction to Machine Learning (`http://www.r2d3.us/visual-intro-to-machine-learning-part-1/`) [2] and the Deep Playground illustration of Tensorflow (`http://playground.tensorflow.org/`) [1]. These d3-based web pages illustrate principles of their respective machine learning algorithms by showing the flow of data through the algorithm. Tensorflow additionally allows the user to alter the model's hyperparameters and then watch the algotithm iterate through time.

These projects inspired the machine learning teaching tool developed herein by expanding its author's understanding of the types of visualizations made possible by D3. They also showed that interactive visualizations can be leveraged to communicate machine learning principles that can be more difficult to communicate with static text and images.

METHODS

Atomic Force Microscopy (AFM) images of perovskite solar cells were provided from David Ginger's lab in the UW Department of Chemistry. This data consisted of arrays of sample height, potential, phase, and amplitude as well as sample photoluminesence. The spatial nature of these arrays makes visualizing them as heatmaps or images a natural choice. The research idea behind the project was to explore if machine learning could be used to predict the photoluminesence (a long experiment) from the height, potential, phase, and amplitude (much easier experiment). This machine learning was done using scikit-learn in Python [4]. Models were trained on 75% of the data (49,152 pixel values for each input), reserving 25% for testing. Generated models varied by two hyperparameters: tree depth, and number of trees in each forest. Both parameters were varied in combinations from one to five; additional depth and trees in each forest did not reduce the mean squared error of the predictions. Each tree in the forests was trained using default parameters, which results in random sampling of the input features with replacement until the original sample size is reached.

The script tree_to_dict.py (`https://github.com/CSE512-16S/fp-wesleybeckner-mdmurbach-janetmatsen/blob/gh-pages/python/scripts/tree_to_dict.py`) was developed to convert the tree structure from the scikit-learn class _tree to D3-ready JSON files. This script recurses over the tree structure from the root, storing decision information and flow through the node in dictionaries. By nesting each child dictionary inside its parent, a tree-like structure was created. These trees were then exported to JSON text files, which were loaded into D3 using built-in d3.json code.

Final Design

The rich information of the decision trees were communicated by the following encodings (Figure 2):

1. **Nodes are colored by the feature used for the decision.** The node colors correspond to the border of pixels when

displayed as enlargements over the four training data images. The domination of blue and orange in the trees communicate the importance of potential and phase in predicting the photoluminesence.

2. **Opacity and stroke weight are used to feature the most important components of the tree**. The number of training points that went through each tree node was obtained from Python. By dividing these counts by the total number of training points, a measure of "flow" through each branch was obtained. These were used to control the transparency of nodes and the thickness of edges.

3. **Hover labels communicate node and leaf information.** Labels showing the decision being made at a node, or the summary of a leaf are displayed upon hover.

Sliders are used to control the hyperparameters of the learning algorithm. When the user varies the slider, a different JSON file corresponding to the correct depth and number of trees is loaded and the images and tree outputs are updated.

To create a mental connection between the input and output features of the images, we appropriated the following elements:

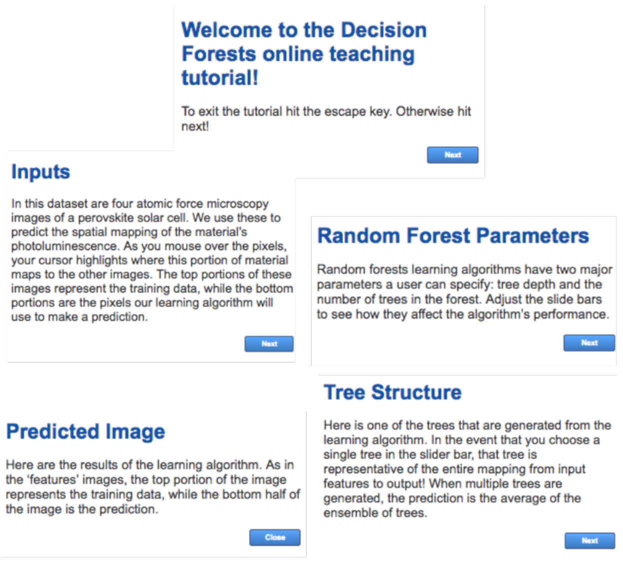
1. **Cursor position is updated across all images and prints pixel values**. As the user mouses over the images, an identical cursor symbol appears in the other images, indicating which features correspond to an area of the material to create a single "feature set" that is fed through the decision tree. In the top left corner of the images, the pixel value is printed.

2. **Magnified pixels have a color-coded border that coordinates with the decision tree coloring**. The border that appears around the cursor is color-coded the same as the decision tree nodes. With the first mouseover event, the user can immediately see which input images (height, potential, phase, and amplitude) dominate the nodes in the decision tree; in this case, potential and phase.

3. **Magnification of the pixels are animated**. In our first iteration of the tutorial design, mousing over the images would render an enlarged pixel. Trial users were unsure what the enlarged pixel was representing. By animating the transition from small to large, the connection became immediately apparent that the cursor was a magnifying tool.

4. **All images use the viridis colormap**. We tried a variety of python colormaps to encode the pixel values in the various AFM images: seismic, spectral, and BrBG from the diverging colormaps; and viridis, inferno, plasma, and magma from the perceptually uniform colormaps. The virdis colormapping reproduced the best contrast within every image—in spite of pixel values ranging across several orders of magnitude.

Lastly, to make the application a stand-alone tutorial, we created a set of guiding boxes that the user can opt out of at any time by hitting the escape key, shown in Fig 1. The windows walk the user between the features, parameters, trees, and outputs elements of the application dashboard.



Welcome to the Decision Forests online teaching tutorial!

To exit the tutorial hit the escape key. Otherwise hit next!

Next

Inputs

In this dataset are four atomic force microscopy images of a perovskite solar cell. We use these to predict the spatial mapping of the material's photoluminescence. As you mouse over the pixels, your cursor highlights where this portion of material maps to the other images. The top portions of these images represent the training data, while the bottom portions are the pixels our learning algorithm will use to make a prediction.

Next

Random Forest Parameters

Random forests learning algorithms have two major parameters a user can specify: tree depth and the number of trees in the forest. Adjust the slide bars to see how they affect the algorithm's performance.

Next

Predicted Image

Here are the results of the learning algorithm. As in the 'features' images, the top portion of the image represents the training data, while the bottom half of the image is the prediction.

Close

Tree Structure

Here is one of the trees that are generated from the learning algorithm. In the event that you choose a single tree in the slider bar, that tree is representative of the entire mapping from input features to output! When multiple trees are generated, the prediction is the average of the ensemble of trees.

Next

Figure 1. Screenshot of tutorial windows to guide the user through the dashboard.

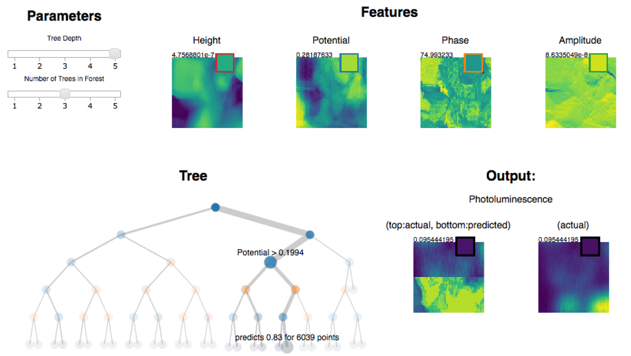


Figure 2. Overview of some interactive components of the visualization. Sliders for the tree depth and number of trees in the forest are shown in the upper left. When hovering over a pixel in an image, the pixel enlarges and the value is displayed as text. The corresponding pixel and values are displayed on all other images. Mousing over the decision tree displays information about nodes. The left output image shows the actual and predicted photoluminescence. The right output image shows the entire actual photoluminescence.

RESULTS

When we shared our visualization with users, the following questions regarding the dataset were raised: why predict photoluminescence, and why is the prediction not more accurate? We were excited to have users become interested in the actual data; it is a success to have the overhead functionality of the application fall into the background and have the real questions regarding the data shine through. Regarding the first question: the photoluminescence AFM image is much harder to measure experimentally than the other AFM images, the goal of the Ginger lab was therefore to gather some intuition about photolumenscence properties given the other AFM images. Our best answer to the second question was that there must be some underlying phenomenon between photoluminescence and one of the input images for there to be an accurate prediction. It may be the case that the other inputs simply cannot help predict photoluminescence. This is an ongoing research question.

Users who were already familiar with machine learning, and especially decision trees, spent on average three minutes learning about the use of machine learning for this application, after an orientation to the components of the graphic and interactive components. Such users enjoyed seeing the final pixel predictions at the leaves of the tree, and an improvement in accuracy as trees and depth were increased.

DISCUSSION

Our tutorial depicts two types of data sets that lend themselves well to visual encoding: decision trees and AFM images. Our task was to determine how to direct focus+context to not overwhelm the user. To this end, for the hyperparameters we appropriated slider bars; for the images we used "viridis" color mapping to differentiate high and low pixel values and tooltips for mapping features together; and for the decision trees colorized nodes indicating feature type being bisected, edge thickness to indicate "feature traffic" through the branch, and tooltips to see the details in the node decision.

We designed the dashboard for both people who don't know any ML and experts. Beginners can learn what a decision tree is, how each split is binary, and how you can use multiple features in one tree. In addition they might note the following anomalies to increase their understanding: sometimes a leaf is created before reaching the final layer (all outputs have been correctly mapped on that edge), some features dominate the binary decision process (these features are physically more indicative of the final output), and input to output traffic can dominate one path of the tree.

Experts can learn how the decision trees perform in the context of this data. We found this to be true by their interest in the underlying data and the aforementioned questions that were posed. Both inexperienced and experienced users appreciated seeing that more possible pixel output values could be predicted given more depth and more trees in the forest.

FUTURE WORK

Display of different trees in the forest

Additional design components were considered. The ability to show different trees from the same forest would help communicate the ensemble aspect of this machine learning strategy, and how the ensemble aspect leads to enhanced performance of forests over trees. Users could discover similarities/differences in the tree architectures among trees in a forest. One possibility would be to show a "carousel" of trees, wherein a user could see an icon representing the different trees in the forest. When an individual tree is clicked on, that tree could be enlarged into the display area currently used for the single tree that is shown.

Animation of the decision tree

One of the most exciting components of the Visual Introduction to Machine Learning website [2] is the point where it shows how features are classified by translating features down the tree (Figure 4).

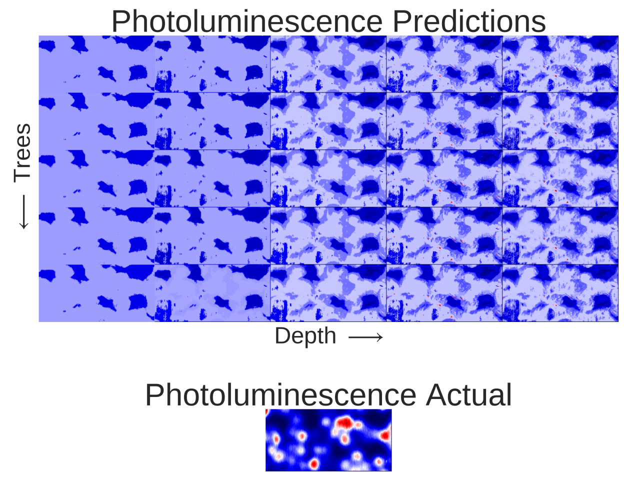


Figure 3. The entire output space available to users of the interactive tutorial displayed in one of our contestant colormaps, the seismic divergent color map. The disparity between low and high values worked extremely well for the photoluminescence images but underperformed viridis in some of the input AFM images.

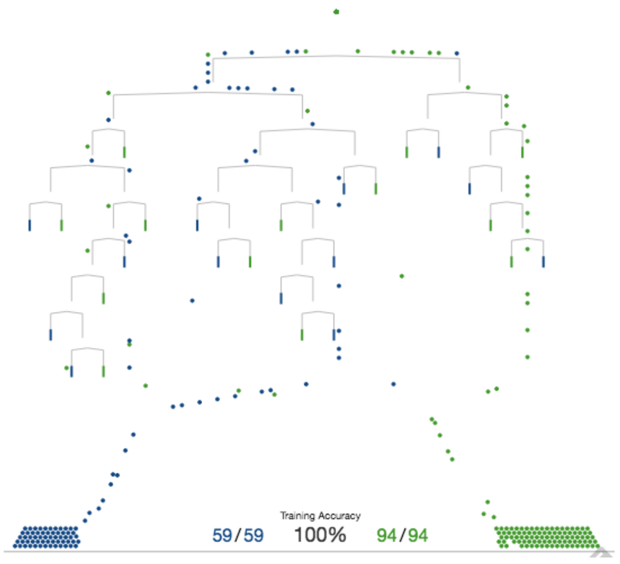


Figure 4. Screenshot of data being classified by the decision tree, as seen in A Visual Introduction to Machine Learning (`http://www.r2d3.us/visual-intro-to-machine-learning-part-1/`)

In order to implement this, one must use the coordinates of the nodes in the tree, and be able to translate a SVG element to the next node based on the feature's properties. The coordinates can be accessed through the D3 tree methods. Correctly moving an input data point through the tree would require javascript functions that can make decisions based on the values of the four pixels and the decision label corresponding to the svg nodes.

Show the feature values predicted at the bottom of the tree

Each leaf's hover text shows the predicted value for training data that ends up at that leaf. This information could also be encoded by appending a correctly colored square just below each leaf to add further visual encoding to the underlying data.

Allow users to upload their own feature vectors and prediction arrays

While this tutorial was developed for AFM images in particular, it could be extended to a wide variety of other use cases. Writing or incorporating a javascript library capable of performing the random forest regression online could enable the full machine learning process to take place in the browser.

REFERENCES

1. Tensor flow illustration. `http://playground.tensorflow.org/`. Accessed: 2016-06-08.
2. A visual introduction to machine learning. `http://www.r2d3.us/visual-intro-to-machine-learning-part-1/`. Accessed: 2016-06-08.
3. Beck, D. A. C., Carothers, J. M., Subramanian, V. R., and Pfaendtner, J. Data science: Accelerating innovation and discovery in chemical engineering. *AIChE Journal 62*, 5 (2016), 1402–1416.
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.