

A Descriptive Framework of Stories of Algorithms

Supplementary Material

J. Liem, R. Henkin, J. Wood, and C. Turkey
City, University of London

Example 1: A visual introduction to machine learning

<http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>

Example 2: The Beginner's Guide to Dimensionality Reduction

<https://idyll-lang.org/gallery/the-beginner-s-guide-to-dimensionality-reduction>

Example 3: t-SNE explained in plain javascript

<https://observablehq.com/@nstrayer/t-sne-explained-in-plain-javascript>

Example 4: Roads from Above

<https://roadsfromabove.netlify.com/>

Legend

Elements of algorithms and models

I Input

L Logic

P Parameters

O Output

Progress depiction

● Snapshot

➔ Continuous

Thematic focus

●/➔ Focus

●/➔ Support

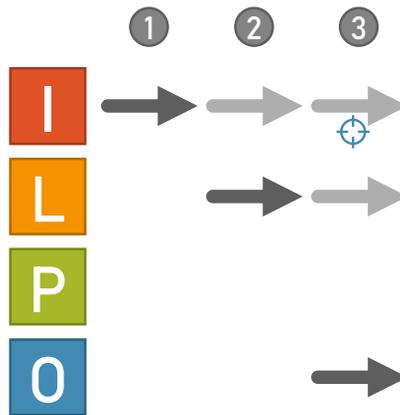
① ② ③ Sequencing of parts of a story

Element linkage

➔ ●
⊕ ⊖ Indication of sources and targets of transformations

A visual introduction to machine learning

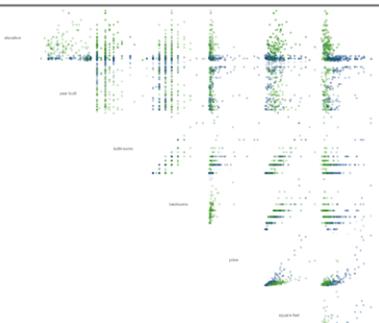
<http://www.r2d3.us/visual-intro-to-machine-learning-part-1/>



1 **I**

scatterplot matrix to show the relationships between each pair of dimensions.

There are clearly patterns in the data, but the boundaries for delineating them are not obvious.



And now, machine learning

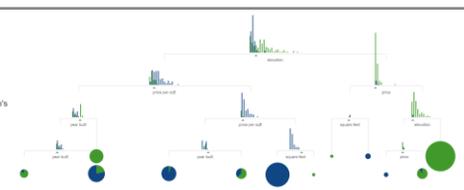
Finding patterns in data is where machine learning comes in. Machine learning methods use statistical learning to identify boundaries.

One example of a machine learning method is a decision tree. Decision trees look at one variable at a time and are a reasonably accessible (though rudimentary) machine learning method.

Explaining input records and features.
The progression is continuously represented.

2 **L**

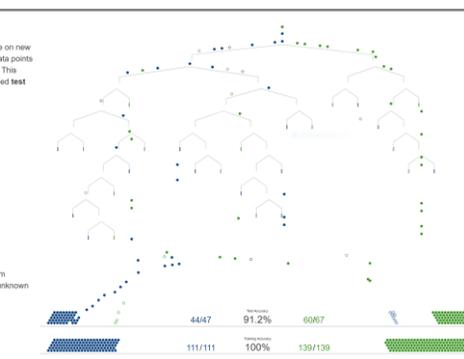
Splitting one layer deeper, the tree's accuracy improves to 84%.



Building the decision tree.
The progression of building the tree is continuously represented. The input has a supporting role.

3 **O**

To test the tree's performance on new data, we need to apply it to data points that it has never seen before. This previously unused data is called **test data**.



Ideally, the tree should perform similarly on both known and unknown data.

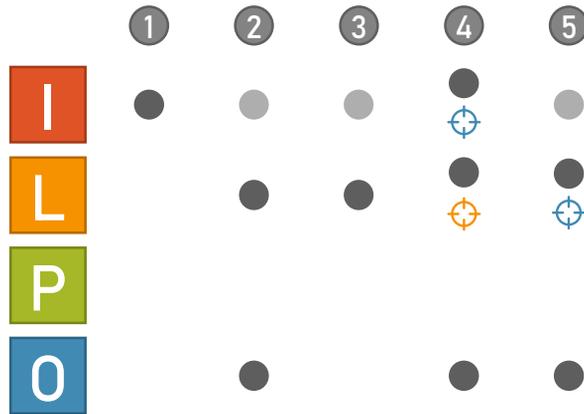
44/47	91.2%	60/67
111/111	100%	139/139

Running the model with a training set and a test set.
Defined by the author, first the user can see how the training set is used to run the model. Then an input change has an effect on the output in an additional model run.

The example follows a logic build up pattern structuring the explanation along the logical order of algorithmic execution.

The Beginner's Guide to Dimensionality Reduction (1)

<https://idyll-lang.org/gallery/the-beginner-s-guide-to-dimensionality-reduction>



1 I

Your browser has just loaded information about roughly 800 artworks from the collection at the Metropolitan Museum of Art. The museum has publicly released a large dataset about their collection^[3], just a small fraction are displayed here. They are positioned randomly.

Hover over an artwork to see its details.

Each artwork includes basic metadata, such as its title, artist, date made, medium, and dimensions. Data scientists like to call metadata for each data point (artwork) *features*. Below are some of the features of 10 artworks in the dataset.

Year	Title	Artist
1680	Spindle-back armc...	
1875	Armchair	Potter and Stymus ...
1776	Basket	Myer Myers
1650	Basin	Master Potter A

Explaining the records and features used as input. A snapshot of the input that will be used in an algorithm is presented.

2 I
L
O

Projecting onto a line

These features can be thought of as vectors existing in a high-dimensional space. Visualizing the vectors would reveal a lot about the distribution of the data, however humans can't see so many dimensions all at once.

Instead the data can be projected onto a lower dimension, one that can be visualized directly. This kind of projection is called an *embedding*.

Computing a 1-dimensional embedding requires taking each artwork and computing a single number to describe it. A benefit of reducing to 1D is that the numbers, and the artworks, can be sorted on a line.

Features

f_1, f_2, f_3, f_4, f_5

Explaining the logic and output of the algorithm. A simple model maps the input (in a supporting role) to the output. The text also includes an overview of the logic used.

3 I
L

For the mathematically inclined

Dimensionality reduction can be formulated mathematically in the context of a given dataset. Consider a dataset represented as a matrix X , where X is of size $m \times n$, where m represents the number of rows of X , and n represents the number of columns.

Typically, the rows of the matrix are *data points* and the columns are *features*. Dimensionality reduction will reduce the number of features of each data point, turning X into a new matrix, X' , of size $m \times d$, where $d < n$. For visualizations we typically set d to be 1, 2 or 3.

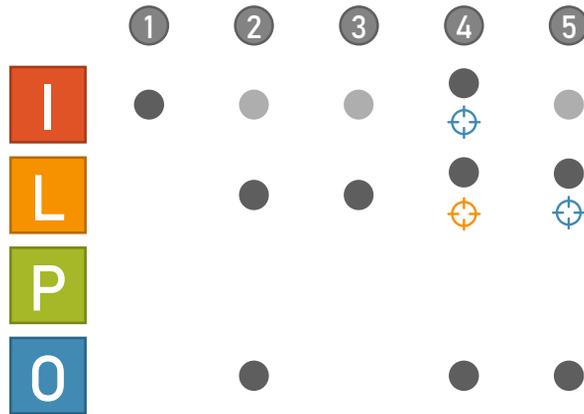
Say $m = n$, that is X is a square matrix. Performing dimensionality reduction on X and setting $d = 2$ will change it from a square matrix to a tall, rectangular matrix.

$$X = \begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix} \implies \begin{bmatrix} x' & x' \\ x' & x' \\ x' & x' \end{bmatrix} = X'$$

Explaining the logic of the algorithm. The input is still present in a supporting role, yet in this case the output is not related to the algorithm. The focus is on the textual explanation of the algorithm.

The Beginner's Guide to Dimensionality Reduction (2)

<https://idyll-lang.org/gallery/the-beginner-s-guide-to-dimensionality-reduction>



4 I L O

Embedding data in two dimensions

The same brightness feature can be used to position the artworks in 2D space instead of 1D. The pieces have more room to spread out.

On the right you see a simple 2-dimensional embedding based on image brightness, but this isn't the only way to position the artworks. In fact, there are many, and some projections are more useful than others.

Use the slider to vary the influence that the brightness and artwork age have in determining the embedding positions. As you move the slider from brightness to artwork age, the embedding changes from highlighting bright and dark images, and starts to cluster recent modern-day images in the bottom left corner whereas older artworks are moved farther away (hover over images to see their date).

Artwork Age Brightness

[SHOW TECHNICAL DETAILS](#)

Explaining the relationship between input and output.

A snapshot of the output can be modified by interacting with the input. Here, an optional interaction stage with the underlying mechanism enables inspecting it (as seen in the orange target above).

5 I L O

PCA t-SNE UMAP

Principal component analysis

Pros:

- Relatively computationally cheap.
- Can save embedding model to then project new data points into the reduced space.

Cons:

- Linear reduction limits information that can be captured; not as discriminably clustered as other algorithms.

There are many algorithms that compute a dimensionality reduction of a dataset. Simpler algorithms such as principal component analysis (PCA) maximize the variance in the data to produce the best possible embedding. More

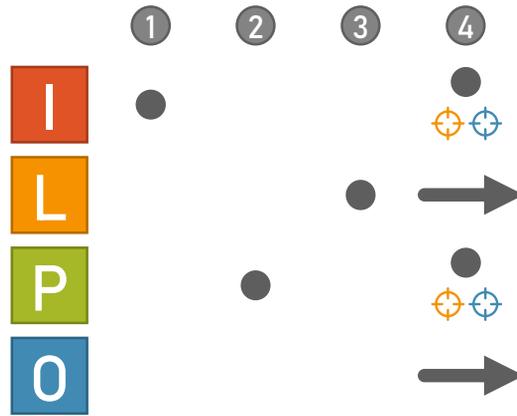
Explaining the logic of the algorithm.

Here it's possible to explore different implementations of dimensionality reduction algorithms. Input is present in a supporting role, whereas interacting with the logic affects the output.

The story has a logic complexity build up pattern, where the complexity of the algorithm increases from 1D, a simple 2D and finally the choice between three implementations.

t-SNE explained in plain javascript (1)

<https://observablehq.com/@nstrayer/t-sne-explained-in-plain-javascript>



1 I

With 50 total datapoints or observations, we can calculate the total number of permutations:

Set number of observations in your dataset

$$n^2 = 50^2 = \boxed{2500 \text{ pairs}}$$

Now if we're efficient about it and only look at the combinations of the data we can cut this down. The combination equation is a bit more complicated...

$$\frac{n!}{2!(n-2)!} = \frac{50!}{2 \cdot 48!} = \frac{\dots 47 \cdot 48 \cdot 49 \cdot 50}{2 \cdot (\dots 47 \cdot 48)} = \frac{49 \cdot 50}{2} = \boxed{1225 \text{ pairs}}$$

So we can see that by taking advantage of this combination vs permutation distinction we are saving ourselves 1275 total distance calculations. Kinda nice, especially when the data gets larger.

Explaining how the input affects the algorithm. An interactive slider helps with a textual explanation.

2 P

What is it?

Perplexity a measure related to the [entropy](#) (or dispersed-ness) of the system of points. A valuable way of thinking about perplexity is to think of it as the effective number of neighbors for each point. Aka how many points that the distribution over point i capture with a non-trivial probability.

The equation is relatively simple. (The calculations for p_{ji} are coming up next, so don't worry about them yet)...

$$\log(\text{perplexity for point } i) = (\text{entropy for point } i) = \sum_{j \neq i} -p_{ji} \log(p_{ji})$$

```
calc_entropy = f(probs)
function calc_entropy(probs){
  return probs.reduce(
    (sum, p) => sum - (p>1e-7 ? p*log(p): 0), // super small probabilities can be ignored
    0
  )
}
```

Explanation of a parameter. A textual snapshot represents the parameter.

3 L

Actually running it

Now that we have functions for calculating the cost and gradient for a given mapping we can (finally) proceed with the actual algorithm.

The steps we need to do for t-SNE are as follows:

1. Randomly initialize mappings mappings: Y_{init}
2. Calculate the cost and gradient for the current mapping
3. Use the gradient to nudge all our mappings to slightly better positions
4. Repeat 2 and 3 until we've lowered our cost function a satisfactory amount.

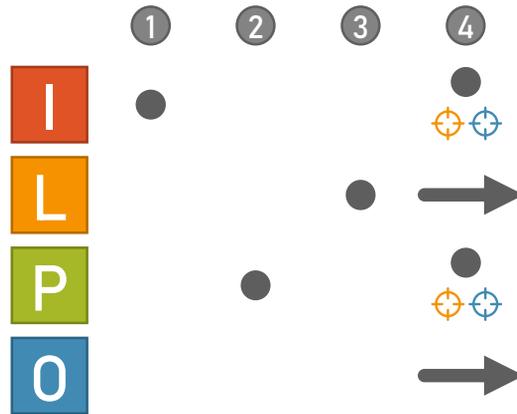
First we initialize a random starting position for the mappings (Y_{new}) and some vectors to help with the gradient descent (which we have not gone into much but it uses [momentum](#) and [gain](#) to increase performance).

We initialize the points purely randomly with a uniform random number generator, but it would be interesting to compare the performance of initializations using different distributions or patterns (such as the [phylotaxis](#) initialization patterns commonly used in [network diagrams](#)).

A textual explanation of the logic of the algorithm.

t-SNE explained in plain javascript (2)

<https://observablehq.com/@nstrayer/t-sne-explained-in-plain-javascript>



4 I L P O

Visualizing the visualizer

We've done everything! Now we can step back and let the algorithm do its thing.

Data
To demonstrate the algorithm I have written a function that generates data in 10 dimensions centered around 5 randomly placed clusters. The points are colored according to which cluster they belong to.

The plots
Below are two plots, the above is the current position of the mappings at the current iteration and below it is the cost function plotted over time. Tinker with all the sliders below to change various aspects of the visualization.

Reset

Iteration: 199

Cost

Perplexity value

Number of iterations to run algorithm

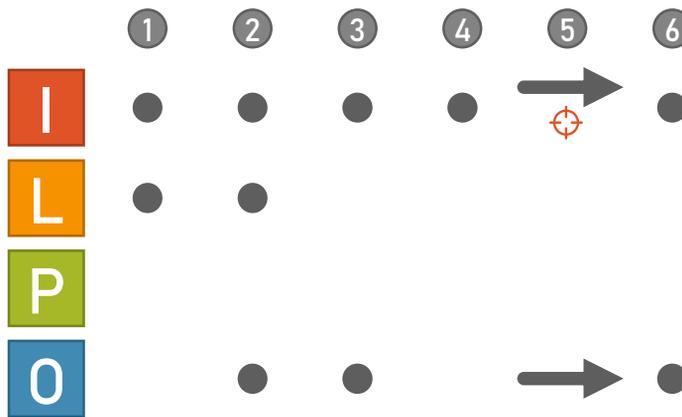
How many iterations to exaggerate

An interactive sandbox combining all elements. Here, input and parameters can be modified with the sliders, while the resulting output is *progressively* shown in the visualization. The lower graphic also contains an output related to the algorithm used – the Cost function.

The story has a build up and sandbox pattern: all the elements are described progressively with snapshots, concluding with a *sandbox* area that allows readers to play around with the elements and see the computation of the results.

Roads from Above (1)

<https://roadsfromabove.netlify.com/>



1 I L

Quantifying the surface area of road networks in cities

This project presents the use of machine learning (using convolutional neural networks) to augment human processes in civil engineering and geospatial applications. In this case an application to quantify the amount of asphalt (pavement) on a defined area of road network.

The process takes aerial images as input and translates these into units & layers of abstracted patterns (called kernels). These patterns become the building blocks to enable the machine learning system to classify every pixel of an image for this project as either road or non-road.

ABSTRACTION

AERIAL PHOTOGRAPHY

Explaining input and intermediate steps of the model. In two visualisations the input imagery can be compared to different layers of the neural network.

2 I L O

Classification (as separate categories road & non-road) is returned as a number between 0 - 1 to indicate the certainty of the result. A value reaching 1 confirms high certainty in the result, whereas a value like .45 less confidence in what has been detected.

Selected pixel from aerial photograph

Use surrounding pixels to create image sample (4x4)

Grouped feature maps from the convolutional blocks

Classifier output as a probability value

0.001

Pixel classification: Non-Road

Algorithm overview.

This part explains the basic idea of the algorithm, including input (photo imagery), logic (convolutional neural network) and the output (classification result).

3 I O

Classifications: non-rd non-rd low certanty

Comparing input and output.

A subset of the input pixels can be compared to their output classification in form of snapshots using an interactive swipe tool.

4 I

Existing road network is used to identify road centers

Pixels in close proximity to centers here are labeled Road

Pixels distant to the centers here are labeled Non-Road

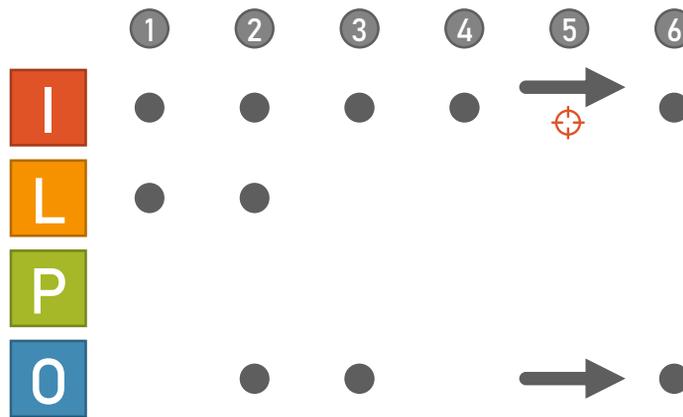
Combined Road & Non-Road pixels used as training data

Explaining training set.

This part explains how the training data was generated and labelled.

Roads from Above (2)

<https://roadsfromabove.netlify.com/>



5 I O

Training & Testing Demonstration

The following demonstration simulates the training and testing process of our model: toggle between training & testing phases, and when testing select different locations to see the different classification results.

19992 labelled pixel samples

Samples for Training

All samples are known & labelled as either **Road** or **Non-road**. These samples - the training data - are selected from a fixed area, or subset, of the overall aerial imagery.

Training Testing

Results

Through iterative testing and evaluation the model is adjusted to give accurate results relative to the known & labelled sample data. An acceptable tolerance of **Low certainty** results are retained for further inspection by expert.

From input to output.

An animation shows which input pixels result in which output class. A user can follow the animation for the training set or can select a new test area interactively. The animation then scans the input imagery from top to bottom selecting sample locations, which move from left to right while being applied to a class (watch online!). Compare to the other parts the story this part is presented in a continuous way.

6 I O

Applying the model to each pixel of the aerial photography reveals an image of landscape with more or less certainty (a pixel-wise probability map) regards to whether the pixel is road or non-road.

Classifications: road non-road low certainty

After applying the model to every pixel of the input aerial images, further image processing techniques (noise removal and image sharpening) can be applied on the output images to remove the isolated points and groupings (e.g. elements not associated with the road network etc.)

Comparing input and output.

Again, using an interactive swipe tool the user can compare the input imagery with the output classification (roads, non-road, low certainty) in the manner of snapshots

The story follows an input-effect pattern, where each part of the story explains how the input influences other elements such as intermediate stages of the model logic or the model output.