Summary

Winnower is a pair of software scripts for measuring and classifying leaves from cherry and pear trees. Winnower comprises of `winnower-size.py` for identifying and measuring objects from scanned images of known resolution, and `winnower-lda.py` for classifying measured objects using linear discriminant analysis. Both scripts provide immediate visual feedback of their progress. Linear discriminant analysis is a statistical method for automatically setting a decision criteria for binary classification (identifying objects as belonging to one of two classes). This report mathematically derives key properties of linear discriminant analysis and provides a flexible software implementation. Winnower can easily be adapted to classify any dataset of scanned two-dimensional objects by modifying the software parameters.

Introduction

Two problems must be solved when classifying objects by their physical properties. First, measurements which are useful for classification must be extracted from the objects. Second, a method for converting those measurements into a classification decision must be chosen.

Measurements should be made in a fast, accurate, and reproducible manner. Additionally, the measurements should differ between each class on average, in order to be useful for classification. Two-dimensional objects can be scanned via a scanner to quickly create accurate and reproducible images of the object, whereas photographs can distort object dimensions. Software analysis of scanned images via computer vision techniques is fast, accurate, and reproducible, unlike manual measurements (e.g. using a ruler) which cost time and may not be reproducible due to human error. Another advantage of software is that it can be upgraded to generate new types of measurements immediately from existing scanned images, whereas additional human measurements would cost time. This allows measurements to be added and modified according to their usefulness to classification. Although software occasionally fails, human review can still be used to find issues. This review is quicker and more reliable than human measurement so long as the software provides feedback in a clear and timely manner.

Classification is traditionally done by humans with expert knowledge, which is neither scalable nor reproducible. Instead, statistical methods can be used to mathematically generate classification rules which apply best to a set of already classified training samples. Many statistical methods guarantee attractive properties such as maximizing the statistical likelihood of the classification, meaning that they produce the most accurate predictions possible given the provided training dataset and statistical model. In addition, these rules are reproducible and easily adapted to new types of measurements or new classification tasks. One drawback is that most statistical approaches need labelled training samples in to develop the classification rules. As a result, new samples which differ significantly from the training samples may not be classified correctly. To avoid this problem, a sufficient number of labelled training samples must be provided which covers the range of variation found in the natural population. This project applies the technique of linear discriminant analysis, which requires a relatively small number of labelled samples for high quality classification (approximately 16 of each class). If labelled samples are not available, such as when discriminating between leaves of two unknown species, other statistical methods can be applied such as a Gaussian mixture model or autoencoder.

Data collection

A Python script called `winnower-size.py` (Appendix D) is developed for automatically measuring scanned images of objects, using the software library OpenCV. This is a command line script which requires the installation of Python 3 and supporting Python packages:

- Python version 3.6 or greater
- Numpy
- Pandas

- Scipy
- OpenCV-Python
- Imutils

The following example commands can be used to install the above in a standard Debian-based Linux system. Warning: these commands have not been tested on your system! Please do not run any commands if unsure of their effect, and contact IT or HPC support to install software instead.

```
sudo apt update
sudo apt install python3 python3-pip
sudo pip3 install --upgrade pip
pip3 install numpy==1.18.4 pandas==1.0.4 scipy==1.5.1 \
    opencv-python==4.4.0.42 imutils==0.5.4
```

`winnower-size.py` has the following parameters:

- `-i`, `--input`: (required) Path to the input directory/folder containing the class subdirectories/subfolders which contain the individual scanned images. Directory structure must be (classes)/(images), and images must be in PNG format.
- `-o`, `--output`: (required) Path to the output directory/folder where annotated images and measurements will be stored. Note: this will overwrite existing images and measurements!
- `-s`, `--scale`: (default 300 DPI to mm) Conversion factor for pixels to desired measurement unit. Set based on scanner calibration as (actual length in desired units) / (length in pixels in image).
- `-m`, `--min_area`: (default 1000 mm^2) Minimum area of valid objects in units squared (conversion according to –scale). Increase if noise is classified as valid objects, decrease if valid objects are being omitted.
- `-M`, `--max_area`: (default 5000 mm^2) Maximum area of valid objects in units squared (conversion according to –scale). Decrease if noise is classified as valid objects, increase if valid objects are being omitted.
- `-r`, `--rotate_steps`: (default 90 gives rotations of one degree) When measuring length and width, this gives the number of rotations of bounding box to test for maximum and minimum bounding box dimensions. Reduce only if computation time is excessively long.
- `-a`, `--angle_dist`: (default 5mm) When measuring angles along perimeter, this gives the minimum distance along perimeter to travel before reporting change in angle. Reduce if measured angles are too coarse, and increase if measured angles are too fine.
- `-d`, `--display`: (default False) When set to `1` or `True`, this will immediately show images in a window as they are processed. Press ESC, CTRL+C, or close window to continue processing. This is useful for debugging or adjusting parameter values.

To run the script, run the following command in the folder/directory containing `winnower-size.py`, replacing the text in square brackets as necessary:

```
python3 winnower-size.py -i [input location] -o [output location] [optional arguments]
```

For example, to run `winnower-size.py` on the folder/directory `indir`, save outputs to `outdir`, use a custom scaling parameter of 0.12, and display each image as it is processed, run:

```
python3 winnower-size.py --input indir --output outdir --scale 0.12 --display True
```

This is identical to:

```
python3 winnower-size.py -i indir -o outdir -s 0.12 -d 1
```

Generating scans

`winnower-size.py` operates on scanned images of the objects to be measured. Images should adhere to the following requirements:

- Scan images in the `.png` format.

- Scan images in black and white format with sufficient contrast so that objects are mostly black and the background is mostly white. The software will ignore small variations and imperfections based on the `--min_area` parameter, and ignore large margins or other processing noise based on the `--max_area` parameter.
- Multiple objects can be scanned in one image so long as sufficient empty space is left between objects.
- Sort images by class into folders/directories which are named according to the class's name. For example, an image of cherry leaves should have the path `data/cherry/cherry1.png`, and an image of pear leaves should have the path `data/pear/pear1.png`
- Any file name can be used for individual images: for example, `image-cherry.png`, `cherry-a.png` are all acceptable names.
- Only one class of leaf/object should be present in each scanned image.

Calibration

To avoid measurement error, either a single scanner or a common scanner make/model should be used to scan all objects. Ensure that the scanner's resolution is set at a fixed level (e.g. 300 DPI) for all scans. Verify the scanner's stated resolution by scanning an object of known length and comparing the scanned size in pixels to the object's true length. To convert between length L, pixels P, and conversion factor:

$$L = PC \implies C = \frac{L}{P}$$

For example, suppose a disc of diameter $100mm$ is scanned as `calibrate.png`. Run `winnower-size.py -i calibrate.png -o calibrate -s 1` to find the disc diameter in pixels as P, then the conversion factor for pixels to millimeters is $C = \frac{100}{P}$. Set the parameters `-s` or equivalently `--scale` to the calculated value of C for all subsequent scans.

Measurements

`winnower-size.py` saves annotated images into the output folder/directory using a file structure identical to the input folder/directory's structure. Each annotated image is named after the original image's file name, but with `labelled` appended to the end of the name (e.g. `cherry1.png` becomes `cherry1labelled.png`). Additionally, measurements are summarized in a CSV file at the root of the output folder/directory in a file called `measurements.csv`. All length measurements are reported in the unit calibrated by the `--scale` parameter, and all angle measurements are reported in degrees. Measurements are rounded to the nearest unit.

The following properties are reported in the CSV file:

- `ID`: An integer starting from 1 which is unique to each identified object across all of the processed images. Use this to cross-reference measurements with the annotated images.
- `CLASS`: The class of the object, corresponding to the name of the subfolder/subdirectory containing the original image.
- `WIDTH`: The smallest dimension possible for a bounding box of arbitrary rotation to enclose the object. In other words, the narrowest possible parallel slit through which the object can fit.
- `LENGTH`: The largest dimension possible for a bounding box of arbitrary rotation to enclose the object. In other words, the widest gap across which the object can span.
- `PERIMETER`: The length of the convex hull enclosing the object. The convex hull is the smallest shape which encloses the object and does not curve inwards towards the object.
- `AREA`: The area of the convex hull enclosing the object. Reported in units squared.
- `ANGLE1`: The most acute angle in the convex hull, calculated as the total change in direction over a minimum distance along the perimeter of the convex hull.
- `ANGLE2`: The second most acute angle in the convex hull.

Examples of annotated images and the generated CSV file are in Appendix A.

Analysis

A Python script called `winnower-lda.py` (Appendix D) is developed to automatically classify the `measurement.csv` files generated by `winnower-size.py` using linear discriminant analysis.

`winnower-lda.py` has the following parameters:

- `-t`, `--train`: (required) Location of training data in the form of a CSV file generated by `winnower-size.py`.
- `-T`, `--test`: (optional) If supplied, the location of test data in the form of a CSV file. The test data does not require a `CLASS` column, and can contain any nonempty subset of the measurements provided to `--train`. The script will automatically remove any columns which are not common to both the `--train` and `--test` files.
- `-o`, `--output`: (optional) Path to the output file where test results will be stored. If `CLASS` is provided, the script will also calculate the accuracy of the generated predictions.
- `-c`, `--covariance`: (default `False`) When set to `1` or `True`, calculate covariance matrix separately for each class. Otherwise, use the pooled covariance matrix for all classes (weighted average by the number of samples in each class) for classification.
- `-d`, `--display`: (default False) When the number of features is 2 and this is set to `1` or `True`, a plot of the data points and decision boundary will be displayed. If `--output` is set, the plot will be saved to a PNG file with the same name.
- `-m`, `--margin`: (default 0.2) The width of the margin around plotted points, as a proportion of the extent of the points.
- `-r`, `--resolution`: (default 100) Resolution of the density and decision boundary plots.

To run the script, run the following command in the folder/directory containing `winnower-lda.py`, replacing the text in square brackets as necessary:

```
python3 winnower-lda.py -t [input location] [optional arguments]
```

For example, to run `winnower-lda.py` using the training data in `outdir/measurements.csv`, testing on the same data, with separate covariance matrices, plotting the result, and saving to `testresults`, run:

```
python3 winnower-lda.py --train outdir/measurements.csv --test outdir/measurements.csv \
    --output testresults --covariance True -display True
```

This is identical to:

```
python3 winnower-lda.py -t outdir/measurements.csv -T outdir/measurements.csv \
    -o testresults -c 1 -d 1
```

Linear Discriminant Analysis

A brief overview of linear discriminant analysis (LDA) is provided here, with full derivations in Appendix B. A sample $\mathbf{x} = (x_1, x_2, \dots x_k)$ consists of k measurements such as `WIDTH`, `LENGTH`, `PERIMETER`, etc., which are assumed to be multivariate normal conditional on the sample's class, with each class having mean vector $\mu = (\mu_1, \dots, \mu_k) \in \mathbb{R}^k$ and covariance matrix $\mathbf{\Sigma} = \left[\sigma_{ij}\right] \in \mathbb{R}^{k \times k}$. The parameters for the sample distribution are estimated via method of moments as:

$$\hat{\mu}_i = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad \hat{\sigma}_{ij} = \left(\frac{1}{n} \sum_{i=1}^{n} x_i x_j \right) - \hat{\mu}_i \hat{\mu}_j$$

where n is the number of samples. To classify a sample $\mathbf{x}$, let

$$LDA(\mathbf{x}) = \begin{cases} \text{class a}, & \lambda > 1 \\ \text{class b}, & \lambda < 1 \\ \text{unclassified}, & \lambda = 1 \end{cases} \qquad \lambda := \frac{p(\mathbf{x}|\hat{\mathbf{a}}, \hat{\mathbf{\Sigma}}_a)}{p(\mathbf{x}|\hat{\mathbf{b}}, \hat{\mathbf{\Sigma}}_b)}$$

where $\hat{\mathbf{a}}$ and $\hat{\mathbf{\Sigma}}_a$ are the estimated mean and covariance for class a's distribution (similarly $\hat{\mathbf{b}}$ and $\hat{\mathbf{\Sigma}}_b$ are the estimated mean and covariance for class b).

In theory, the set of unclassified points has measure zero for non-degenerate distributions of a and b, meaning all points should be classified as a or b. In practice, numerical issues such as rounding may cause unclassified points, which `winnower-lda.py` solves by assigning $\lambda = 1$ to class b.

The decision boundary is the set of points where $\lambda = 1$. Although it is not necessary to know the decision boundary to classify points, we examine its geometry in the case where the covariance matrices of each class are assumed to be equal, and the case where they are assumed to differ.

Pooled Covariances

If the two class distributions differ only in their means, let $\hat{\mathbf{\Sigma}}$ be the common covariance matrix estimated as an average over each class's estimated covariance matrix:

$$\hat{\mathbf{\Sigma}} = \frac{1}{n_a + n_b}\left(n_a\hat{\mathbf{\Sigma}}_a + n_b\hat{\mathbf{\Sigma}}_b\right)$$

where n_a is the number of samples in class a (similarly n_b the number of samples in class b). The decision boundary in this case corresponds to all $\mathbf{x} \in \mathbb{R}^k$ satisfying:

$$\mathbf{r}^T\mathbf{x} - c = 0 \qquad \text{where} \qquad \mathbf{r} = \mathbf{\Sigma}^{-1}(\mathbf{a} - \mathbf{b}) \qquad \text{and} \qquad c = \frac{1}{2}\mathbf{r}^T(\mathbf{a} + \mathbf{b})$$

which is a hyperplane (see Appendix B for explicit solution). In $\mathbb{R}^2$, the boundary is a straight line. By default, `winnower-lda.py` uses this formulation of LDA.

Class-based Covariances

If the class distributions have differing covariances, the decision boundary consists of all $\mathbf{x} \in \mathbb{R}^k$ satisfying:

$$\mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{x}^T\mathbf{b} + c = 0 \qquad \text{where}$$
$$\mathbf{A} = \mathbf{\Sigma}_a^{-1} - \mathbf{\Sigma}_b^{-1} \qquad \mathbf{b} = \mathbf{\Sigma}_a^{-1}\mathbf{a} - \mathbf{\Sigma}_b^{-1}\mathbf{b} \qquad c = \mathbf{a}^T\mathbf{\Sigma}_a^{-1}\mathbf{a} - \mathbf{b}^T\mathbf{\Sigma}_b^{-1}\mathbf{b} + \log(\det\mathbf{\Sigma}_a) - \log(\det\mathbf{\Sigma}_b)$$

which is a conic section (see Appendix B for computational solution via the Newton-Raphson method). In $\mathbb{R}^2$, the boundary is a parabola (line), ellipse (circle), or hyperbola (two crossed lines). To use this formulation of LDA, set the `--covariance` or `-c` flag to `1` or `True`.

Results

Annotated images and measurements for a set of 32 training examples are available in Appendix B.

From visual inspection, cherry leaves tend to be narrower relative to their length than pear leaves. Pear leaves tend to only have one sharp point at their apex, whereas cherry leaves are pointed at both their apex

and base. We define "pointedness" as the angle over which the edge of the leaf rotates in direction over a small distance, which is recorded by Winnower as the two most acute of such angles for each leaf.

Table 1 lists different combinations of measurements and their classification accuracy when applied to the set of 32 training examples.

Table 1: Classification accuracy of feature combinations.

Features	Covariances	Training accuracy %
width, length	Pooled	88
width, length	Separate	88
perimeter, area	Pooled	78
perimeter, area	Separate	88
2 most acute angles	Pooled	97
2 most acute angles	Separate	100
All features	Pooled	97
All features	Separate	100

Among all pairs of measurements, pointedness (2 most acuate angles) provides the most accurate classification. Similar accuracy is achieved by using all features. The use of separate covariance matrices for each class (setting `--covariance` option in `winnower-lda.py`) increases classification accuracy slightly.

Figures 1 and 2 show the decision boundary using pointedness as the classifying feature:

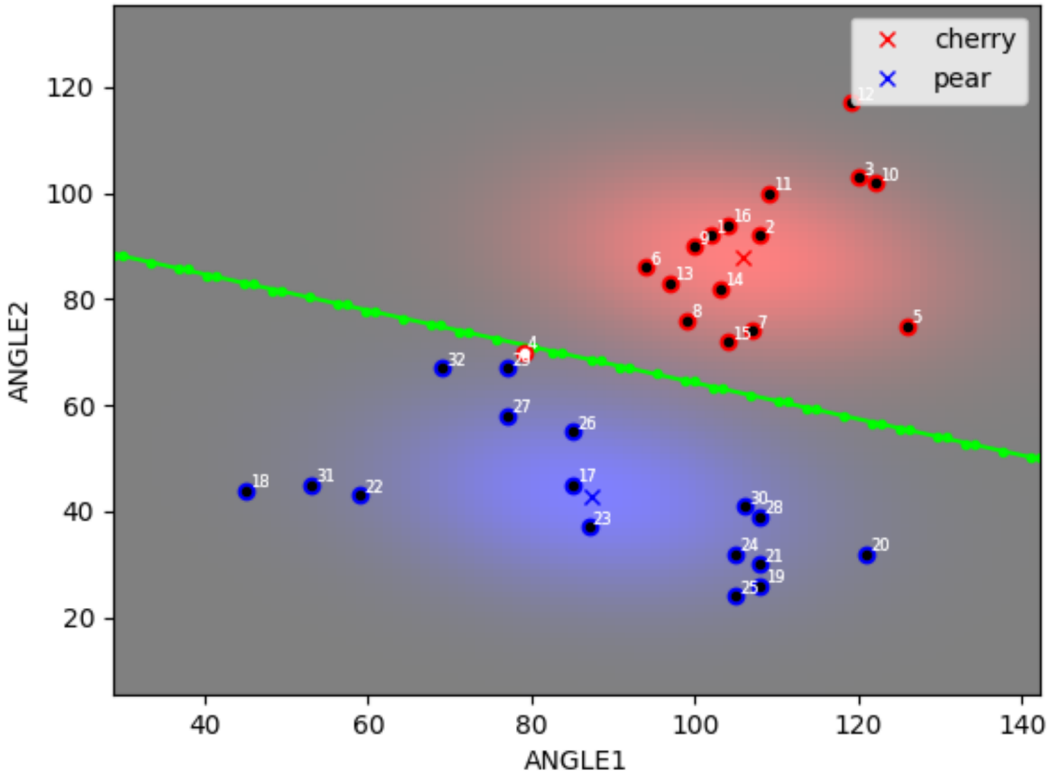

Figure 1: Pointedness classification with pooled covariance (linear decision boundary in green). `ANGLE2` is the most acute angle, and `ANGLE2` is the second most acute angle. Mis-classified points have a white center.

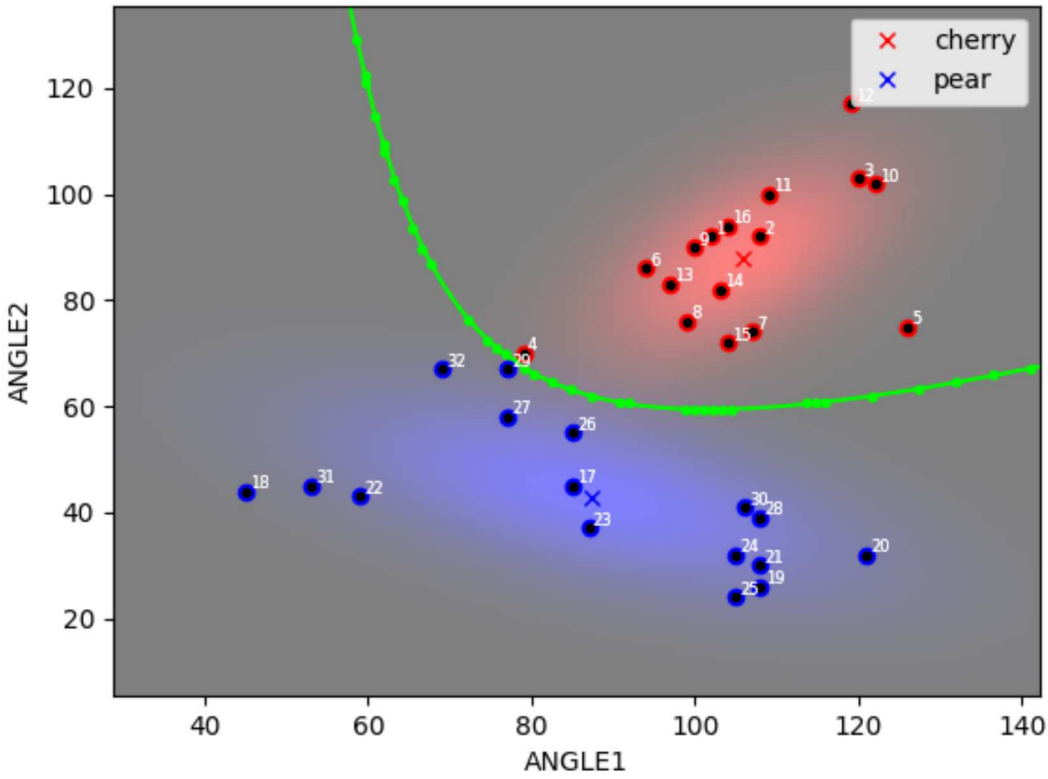

Figure 2: Pointedness classification with separate covariance (quadratic decision boundary in green). `ANGLE2` is the most acute angle, and `ANGLE2` is the second most acute angle. All points are classified correctly.

Small leaves are the most difficult to classify since they are similar in both pointedness and aspect ratio. In particular, the leaves `cherry 2`, `cherry 4`, `pear 26`, and `pear 32` are often mis-predicted.

An additional three leaves $u = (32, 82), v = (38, 52), w = (40, 76)$ are classified with the given width and length measurements. These points are plotted in figures 3 and 4.

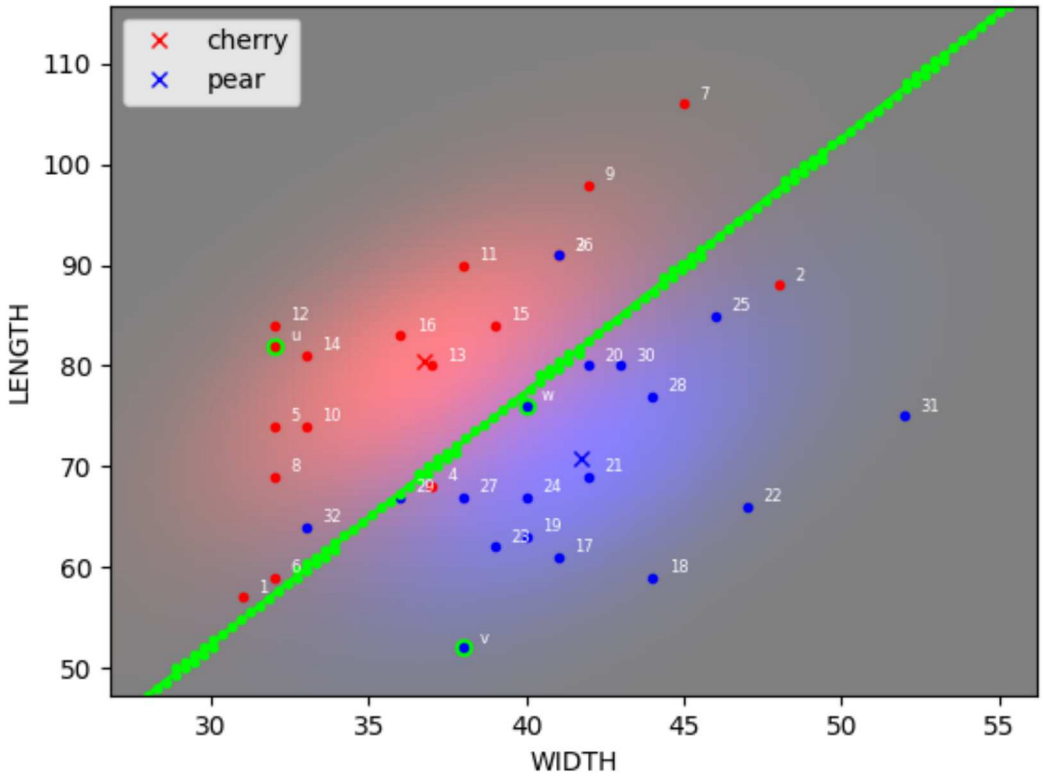

Figure 3: Test points classified using width and length, with pooled covariance (linear decision boundary in green). Test points have a green border, with center color corresponding to classification.

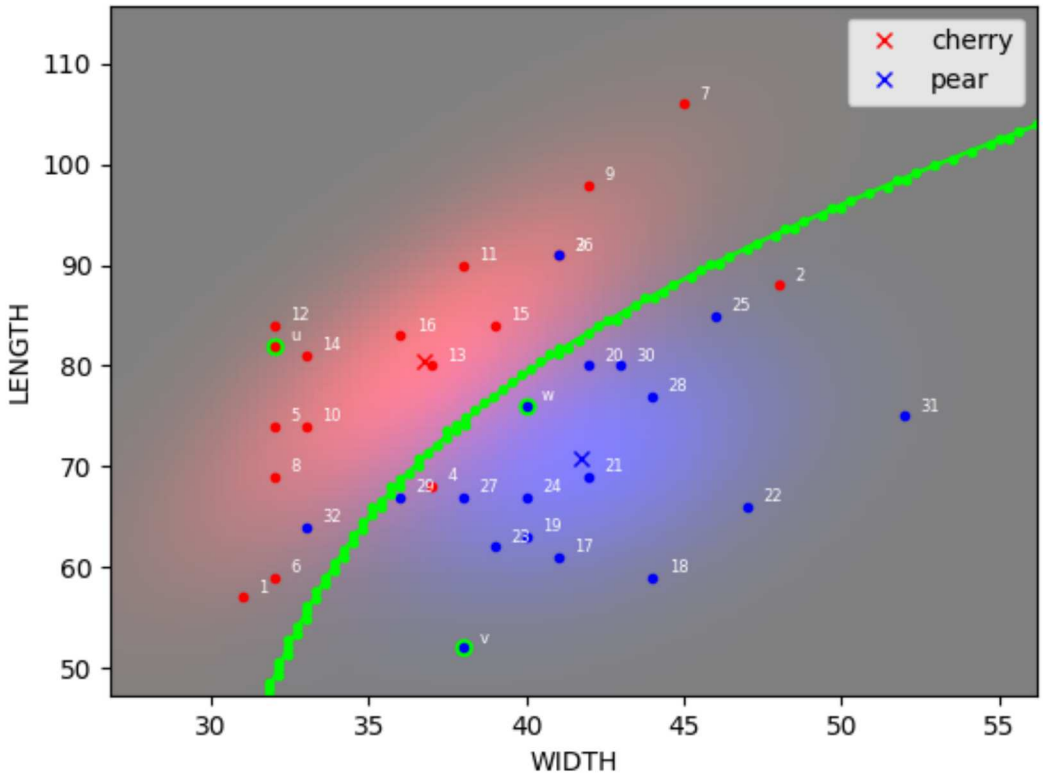

Figure 4: Test points classified using width and length, with separate covariance (quadratic decision boundary in green). Test points have a green border, with center color corresponding to classification.

A full list of results including all classifications, distribution parameters, and decision boundary formulas is in Appendix C.

Conclusion

Winnower is fast, scalable, and reproducible, and can readily be adapted to any two-dimensional object measurement and classification task. In the specific case of classifying cherry and pear leaves, the pointedness of leaves as measured in their two most acute angles is the single most effective pair of measurements for classification. However, since Winnower automatically adapts to any combination of features in the training and testing set, It is recommended to use Winnower in its default configuration, which utilizes all available measurements for classification.